Exhibit 2

Paper No. __

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner,

v.

SINGULAR COMPUTING LLC,
Patent Owner.

Case No. IPR2021-00165
Patent No. 9,218,156

PETITION FOR INTER PARTES REVIEW UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 et seq

TABLE OF CONTENTS

TABLE OF AUTHORITIES

CASES

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 9,218,156
1002	Prosecution History of U.S. Patent No. 9,218,156
1003	Declaration of Richard Goodin
1004	Curriculum Vitae of Richard Goodin
1005	U.S. Patent Appl. 12/816, 201 ("'201 Application")
1006	U.S. Patent Appl. Publ. No. 2010/0325186 A1 ("Bates-2010")
1007	U.S. Patent App. Publ. No. 2007/0203967 ("Dockser")
1008	Tong et. al, *Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 3, June 2000 ("Tong") (from pages 6-19 of the Declaration of Gerard P. Grenier, Ex. 1025).
1009	U.S. Patent No. 5,689,677 ("MacMillan")
1010	U.S. Patent Appl. Publ. No. 2007/0266071 ("Dockser-Lall")
1011	U.S. Patent No. 6,065,209 ("Weiss")
1012	Gaffar et. al, *Unifying Bit-width Optimization for Fixed-Point and Floating-Point Designs*, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, April 20-23, 2004 ("Gaffar") (from pages 22-31 of the Declaration of Gerard P. Grenier, Ex. 1028)
1013	European Patent Appl. Publ. No. 0 632 369 A1 ("Hekstra")
1014	U.S. Patent No. 5,375,084 ("Begun")
1015	U.S. Patent No. 4,933,895 ("Grinberg")
1016	U.S. Patent No. 5,442,577 ("Cohen")
1017	U.S. Patent Appl. Publ. No. 2003/0028759 ("Prabhu")
1018	U.S. Patent No. 5,790,834 ("Dreyer")
1019	U.S. Patent Appl. Publ. No. 2009/0066164 ("Flynn")
1020	U.S. Patent No. 5,666,071 ("Hawkins")
1021	A Matter of Size: Triennial Review of the National Nanotechnology Initiative (National Academies Press 2006), pages 15-44, 99-109
1022	Transcript of YouTube video on Practical Approximate Computing at University of California, Berkeley, March 2016 ("Bates transcript"), video available at https://www.youtube.com/watch?v=aHkWX3QctkM (last accessed Sep. 16, 2020)
1023	U.S. Patent No. 6,311,282 ("Nelson")
1024	U.S. Patent No. 4,583,222 ("Fossum")

1025	Declaration of Gerard P. Grenier regarding Tong et. al, *Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 3, June 2000
1026	Moshnyaga, *Energy Reduction in Queues and Stacks by Adaptive Bitwidth Compression*, Proceedings of the 2001 International Symposium on Low Power Electronics and Design, Aug. 6-7, 2001 ("Moshnyaga"), with IEEE Xplore information appended
1027	Simunic, *Energy-Efficient Design of Battery-Powered Embedded Systems*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, No. 1, Feb. 2001 ("Simunic"), with IEEE Xplore information appended
1028	Declaration of Gerard P. Grenier regarding Gaffar et. al, *Unifying Bit-width Optimization for Fixed-Point and Floating-Point Designs*, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, April 20-23, 2004
1029	U.S. Patent Appl. Publ. No. 2007/0033572 ("Donovan")
1030	U.S. Patent No. 5,623,616 ("Vitale")
1031	David A. Patterson and John L. Hennessy, *Computer Organization and Design* (Morgan Kaufmann 3rd ed. Revised 2007) ("Patterson"), pages 189-217
1032	First Amended Complaint in *Singular Computing LLC v. Google, LLC*, 1-19-cv-12251 (D. Mass.) (Dkt. No. 37)
1033	Docket Report for *Singular Computing LLC v. Google, LLC*, 1-19-cv-12551 (D. Mass.)
1034	Memorandum in Support of Motion to Dismiss for Failure to State a Claim (Dkt. No. 41)
1035	Plaintiff's Opposition to Defendant's Rule 12(b)(6) Motion to Dismiss for Lack of Patentable Subject Matter, *Singular Computing LLC v. Google, LLC*, 1-19-cv-12251 (D. Mass.), Dkt. No. 44
1036	Memorandum and Order on Defendant's Motion to Dismiss (Dkt. No. 51)
1037	Plaintiffs' Preliminary Patent-Related Disclosures Pursuant to Patent L.R. 16.6(d)(1)(A)
1038	Scheduling Order for *Singular Computing LLC v. Google, LLC*, 1-19-cv-12551 (D. Mass.) (Dkt. No. 59)
1039	Transcript of Scheduling Conference held on July 24, 2020

1040	General Order 20-21, Second Supplemental Order Concerning Jury Trials and Related Proceedings, *In Re: Coronavirus Public Emergency* (D. Mass. May 27, 2020)
1041	Letter Re Reservation of Rights
1042	Prosecution History of U.S. Patent No. 8,150,902, issued from U.S. Patent Appl. No. 12/816,201
1043	Prosecution History of U.S. Patent No. 8,407,273
1044	Prosecution History of U.S. Patent No. 10,416,961
1045	Prosecution History of U.S. Patent No. 9,792,088
1046	Prosecution History of U.S. Patent No. 10,120,648
1047	Thomas Way et. al, *Compiling Mechanical Nanocomputer Components*, Global Journal of Computer Science and Technology, Vol. 10, Issue 2 (Ver. 1.0), April 2010, pp. 36-42 ("Way")
1048	David Nield, *In a Huge Milestone, Engineers Build a Working Computer Chip out of Carbon Nanotubes*, Sciencealert.com, Dec. 7, 2019 (accessed Sep. 9, 2020) ("Nield")
1049	Leah Cannon, *What Can DNA-Based Computers Do?*, MIT Technology Review, Feb. 4, 2015 (accessed Sep. 8, 2020) ("Cannon")
1050	Katherine Bourzac, *The First Carbon Nanotube Computer*, MIT Technology Review, Sep. 25, 2013 (accessed Sep. 8, 2020) ("Bourzac")
1051	Joe Touch et. al,, *Optical Computing*, Nanophotonics 2017 6(3): 503-505 ("Touch")
1052	Robert Allen, Ed., *The Penguin Complete English Dictionary*, (Penguin 2006) ("*Penguin*"), page 1411, definition of "supercomputer"
1053	*A Dictionary of Computing* (Oxford 6th ed. 2008) ("*Oxford*"), page 500, definition of "supercomputer"
1054	U.S. Patent Appl. Pub. No. 2006/0270110 ("Steffen")
1055	U.S. Patent Appl. Pub. No. 2009/0188705 ("Kacker")
1056	U.S. Patent Appl. Pub. No. 2007/0050566 ("Lang")
1057	U.S. Patent No. 6,622,135 ("Tremiolles")
1058	U.S. Patent Appl. Pub. No. 2005/0235070 ("Young")
1059	U.S. Patent No. 7,301,436 ("Hopper")
1060	U.S. Patent Appl. Pub. No. 2003/0204760 ("Youngs")
1061	U.S. Patent No. 10,416,961
1062	U.S. Patent No. 8,407,273

MANDATORY NOTICES

A. Real Party-In-Interest

Petitioner is the Real Party-in-Interest.

B. Related Matters

A decision in this proceeding could affect or be affected by the following:

1. United States Patent & Trademark Office

The application from which U.S. Patent 9,218,156 ("the '156 patent") issued is a continuation of U.S. Patent Application No. 13/399,884 (U.S. Patent No. 8,407,273), which is a continuation of U.S. Patent Application No. 12/816,201 (U.S. Patent No. 8,150,902), which claims priority to U.S. Provisional Application No. 61/218,691.

The following U.S. patent applications also claim the benefit of the priority of the filing date of U.S. Provisional Application No. 61/218,691: 14/976,852; 15/784,359; 16/175,131; 16/571,871; 16/675,693; 16/882,686; 16/882,694; and 17/029,780.

2. United States Patent Trial and Appeal Board

Concurrently with the present Petition, Petitioner is filing a second petition challenging claims 1-8, 16-25, and 33-42 of the '156 patent. Petitioner requests that these two petitions challenging the '156 patent be reviewed by the same panel.

Petitioner also previously filed petitions for *inter partes* review of related U.S. Patent No. 10,416,961 under case nos. IPR2021-00154 and IPR2021-00155, and will be filing petitions for *inter partes* review of related U.S. Patents No. 8,407,273. Petitioner requests that these additional petitions also be reviewed by the same panel as the present Petition, as the patents are commonly assigned and share the same specification, and there are numerous overlapping elements among the claims of all three patents.

3. U.S. District Court for the District of Massachusetts

Singular Computing LLC v. Google LLC, Case No. 1:19-cv-12551-FDS.

C. Counsel and Service Information - §§ 42.8(b)(3) and (4)

Lead Counsel	Elisabeth H. Hunt, Reg. No. 67,336
Backup Counsel	Richard F. Giunta, Reg. No. 36,149 Anant K. Saraswat, Reg. No. 76,050 Nathan R. Speed (*pro hac vice* forthcoming)
Service Information	E-mail: EHunt-PTAB@wolfgreenfield.com RGiunta-PTAB@wolfgreenfield.com ASaraswat-PTAB@wolfgreenfield.com NSpeed@wolfgreenfield.com Post and hand delivery: Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 Telephone: 617-646-8000 Facsimile: 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

Petitioner requests *inter partes* review and cancellation of claims 1-8, 16, and 33 ("challenged claims") of the '156 patent (Ex. 1001).

I. INTRODUCTION

The '156 patent claims a device comprising at least one "low precision high dynamic range (LPHDR) execution unit" adapted to execute an operation (*e.g.*, multiplication) on an input signal representing a first numerical value to produce an output signal representing a second numerical value. The "execution unit" is characterized as "LPHDR" because it performs a low precision ("LP") operation on a high dynamic range ("HDR") of values. Goodin Declaration (Ex. 1003, "Goodin"), ¶ 20.

"Dynamic range" refers to the range of numerical values supported. For example, in a binary (base-2) representation of integers, the more bits used, the wider the possible dynamic range—*e.g.*, two bits can represent numbers 0-3, three bits can represent numbers 0-7, etc. Other binary representations (*e.g.*, floating-point) have wider dynamic range for the same number of bits. Goodin, ¶¶ 24-26; '156 patent, 5:28-33; Tong (Ex. 1008), 274.

Precision relates to accuracy and can be impacted by the ability to represent numbers that differ from each other by small amounts—*e.g.*, fractional values. For example, the decimal number 2.13 is more precise than if rounded to 2.1, which is more precise than if rounded to 2. In a binary "fixed-point" representation, any fixed number of bits can be chosen to represent a number's fractional component; *e.g.*, three fractional bits can represent numbers to the nearest 1/8, two fractional bits can represent numbers to the nearest 1/4, and a representation that allocates no bits to the fractional component can only represent integers. Goodin, ¶¶ 21-22. A "low precision arithmetic" operation is less precise than an exact calculation, such that "each [low-precision] operation might introduce" some "error" in its "results" (*e.g.*, by producing a result of 2.1 instead of an exact calculation of 2.13). '156 patent, 4:13-16; Goodin, ¶ 23.

It was known that many numerical representations require a tradeoff between dynamic range and precision. Goodin, ¶ 27. For example, a ten-bit binary fixed-point representation that allocates all ten bits to the non-fractional part (thus representing integers between zero and 1,023) has higher dynamic range but lower precision than a representation that allocates all ten bits to the fractional part (thus only representing values between zero and one but accurate to the nearest 1/1024). Goodin, ¶ 27. Another example relates to well-known floating-point representations, which can represent a high dynamic range using fewer digits than a fixed-point representation—*e.g.*, by representing a large number as a smaller number (the "mantissa") scaled by an order of magnitude. A simple floating-point example using base ten represents the decimal number 3,046,000 as 3.046×10^6. Goodin, ¶ 28.

In a typical binary (base-2) floating-point representation, one bit indicates the number's sign (positive or negative), some bits are allocated to specify the "exponent" (the number's order of magnitude in base two), and other bits are allocated to specify the mantissa. The number of exponent bits impacts dynamic range, the number of mantissa bits impacts precision, and there was a known tradeoff:

> [A] floating-point representation must find a compromise between the size of the [mantissa] fraction and the size of the exponent because a fixed word size means you must take a bit from one to add a bit to the other. This trade-off is between precision and range: increasing the size of the [mantissa] fraction enhances the precision of the fraction, while increasing the size of the exponent increases the range of numbers that can be represented.

Ex. 1031, 191; Goodin, ¶¶ 29-30.

Benefits of LPHDR arithmetic were known in mathematics and computing. For example, Tong (Ex. 1008) recognized "wide dynamic range" as "a desirable feature," and explained "[i]t has long been known that many… applications can get by with less precision." Tong, 273; Goodin, ¶ 31. Tong demonstrated that certain applications could function properly with low-precision HDR arithmetic ("fine precision… is not essential"), and that lowering precision saved power by "reduc[ing] waste [from] unnecessary bits." Tong, 273, 277-279 ; Goodin, ¶¶ 32-34. Similarly, Dockser (Ex. 1007) disclosed a low-precision HDR execution unit that saved power by reducing precision in its floating-point operations to whatever precision was needed for a particular application. Dockser, [0003]-[0007]; Goodin, ¶ 35.

The challenged claims encompass this prior-art concept. Claim 1 recites a device comprising an LPHDR execution unit adapted to execute an operation on an input signal representing a first numerical value to produce an output signal representing a second numerical value, and a generic computing device adapted to control operation of the execution unit. Nothing more. The remainder of the claim simply characterizes the execution unit's performance by reciting its minimum dynamic range and degree of imprecision. Goodin, ¶¶ 36, 38.

II. SUMMARY OF GROUNDS

The challenged claims would have been obvious under § 103 as the following grounds demonstrate. Each reference below (none of which was before the examiner) is prior art under pre-AIA § 102(b) even assuming the challenged claims were entitled to their earliest claimed priority date (they are not as Petitioner's concurrently filed petition demonstrates).

Ground Number and Reference(s)		**Claims**
1	Dockser (Ex. 1007)	1-2, 16
2	Dockser, Tong (Ex. 1008)	1-2, 16, 33
3	Dockser, MacMillan (Ex. 1009)	1-8, 16
4	Dockser, Tong, MacMillan	1-8, 16, 33

Ground 1: Dockser discloses a "floating-point processor" (FPP) that performs "multiplication" at a selectable "precision." Dockser, Abstract, [0012]. Dockser's FPP is an HDR execution unit whose standard floating-point inputs exceed the claimed minimum dynamic range. *Infra* § V.B.3. Recognizing that some applications do not require full-precision operations, Dockser's FPP operates at a selectable reduced precision to conserve power in applications where greater precision is unnecessary. A selected "subprecision" is achieved by removing power to any desired number of least-significant mantissa bits, dropping those bits (resulting in less precision) and reducing power consumption. Dockser, [0014]; Goodin, ¶¶ 375-376. Dockser discloses an example that drops all but the 9 most-significant bits, resulting in imprecision meeting the claimed minimum error amounts. *Infra* § V.B.4. Dockser renders obvious claims 1-2 and 16.

Ground 2: Tong teaches reducing the number of mantissa bits to conserve power, and discloses experimental results demonstrating the optimum balance of precision and power consumption is achieved using just 5 mantissa bits for certain applications. *Infra* § VI.A. The Dockser/Tong combination, in which Tong's optimized precision levels are used in Dockser's LPHDR execution unit, renders obvious the same claims rendered obvious by Ground 1, and meets the claimed minimum error amounts by even greater margins. Tong teaches to emulate in software a device comprising an LPHDR execution unit; thus Dockser/Tong also meets claim 33.

Ground 3: MacMillan discloses a computer system with multiple floating-point execution units operating in parallel. *Infra* § VII.A. Based on MacMillan, a POSA would have been motivated to implement a device with multiple Dockser FPPs operating in parallel. The resulting device (Dockser/MacMillan) meets dependent claim 3's requirement of multiple LPHDR execution units, and provides an alternative basis for meeting the "plurality of components" in independent claim 16. MacMillan's "CPU" that controls the execution units (MacMillan, 10:27-53, 13:12-13, 13:38-62) provides an alternative basis for meeting claim 1's "computing device adapted to control the… execution unit" and claim 2's "CPU." *Infra* §§ VII.B-H.

Ground 4: It would have been obvious to implement the Dockser/MacMillan device with the FPPs operating at Tong's precision levels. This Dockser/Tong/MacMillan combination meets the same claims met collectively by Grounds 1-3.

III. STANDING

Petitioner certifies the '156 patent is available for IPR and Petitioner is not barred or estopped from requesting IPR of the challenged claims. 37 C.F.R. § 42.104(a).

IV. '156 PATENT

A. Challenged Claims

This petition challenges independent claims 1, 16, and 33, and certain dependent claims. Claim 1 is reproduced below, annotated to label claim elements.

> [1A1] A device comprising: at least one first low precision high dynamic range (LPHDR) execution unit
>
> [1A2] adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
>
> [1B1] wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and
>
> [1B2] for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input; and
>
> [1C] at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit.

[1A1]-[1A2] recite a device comprising an LPHDR execution unit adapted to execute an operation. [1B1] recites a claimed dynamic range of the inputs, and

[1B2] recites a claimed minimum imprecision of the operation in terms of minimum relative error (Y) produced for a minimum percentage (X) of inputs. [1C]'s generic "computing device" is adapted to control operation of the execution unit. Goodin, ¶ 38.

Independent claims 16 and 33 also recite an LPHDR unit adapted to execute an operation as in [1A1]-[1A2], with dynamic range and imprecision defined by the X and Y percentage ranges in [1B1]-[1B2]. Claims 16 and 33 recite a "plurality of components," including at least one LPHDR unit, and do not include [1C]. Claim 33's preamble differs from that of claims 1 and 16, and recites instructions executable by a processor to emulate a device comprising the claimed components.

The challenged dependent claims recite specific implementation of claim 1's generic "computing device" (claim 2), a minimum number of LPHDR units (claim 3), higher minimum X and/or Y percentages than in [1B2] (claims 4-6), a higher dynamic range than in [1B1] (claim 7), and that the claimed operation is multiplication (claim 8).

B. Person of Ordinary Skill in the Art ("POSA")

This Petition demonstrates the challenged claims' unpatentability even assuming the '156 patent were entitled to a 2009 priority date. A POSA in 2009 would have had at least a bachelor's degree in Electrical Engineering, Computer

Engineering, Applied Mathematics, or the equivalent, and at least two years of academic or industry experience in computer architecture. More education could substitute for experience, and vice versa. Goodin, ¶¶ 43-44. If the Board determines the ’156 patent is only entitled to its 2013 actual filing date, a POSA in 2013 would have had the same or greater level of skill as in 2009, so the challenged claims would have been obvious for the same reasons discussed herein. Goodin, ¶ 45.

C. Claim Construction

Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. § 282(b), and have been given their ordinary and customary meaning as understood by a POSA in accordance with the specification and prosecution history. 37 C.F.R. § 42.100(b).

D. Prosecution History

The examiner allowed the challenged claims, and the claims in all other applications in the priority chain, without substantively discussing any prior art. Ex. 1042, 164-166; Ex. 1043, 164-167; Ex. 1002, 404-406.

V. <u>GROUND 1:</u> CLAIMS 1-2 AND 16 WOULD HAVE BEEN OBVIOUS OVER DOCKSER

The ’156 patent admits LPHDR execution units were known but alleges they were considered “not useful.” ’156 patent, 6:63-7:16; Goodin, ¶ 184. The patent purports to encompass the idea “that LPHDR arithmetic is useful in several important practical computing applications" to save power ('156 patent, 16:27-29, 23:54-67), but the claims are not limited to methods of use in ***particular*** computing applications. Instead they recite a device comprising an LPHDR execution unit, which was known as Dockser demonstrates. Goodin, ¶ 185.

A. Dockser

Dockser (Ex. 1007)[1] discloses performing "mathematical operations" including "multiplication" using a "floating-point processor having a given precision." [0001], Abstract. Goodin, ¶ 186.

"Floating-point" is a number representation having "a sign component, an exponent, and a mantissa." [0001]. The sign indicates whether the number is positive or negative, and the number's absolute value equals the mantissa multiplied by a base raised to the power of the exponent. [0001]. "[F]loating-point" representation was widely used in computing and refers to a binary (base-2) representation unless some other base is expressed (*e.g.*, decimal floating-point); that is how the term is used herein, in the '156 patent and in the prior-art references cited herein. Goodin, ¶¶ 187-188; '156 patent, 2:35-37; Dockser, [0001]; Tong, 274. As an example, the floating-point representation of the number 6 is: positive sign, exponent=2, mantissa=1.5 (*i.e.*, $1.5 \times 2^2 = 6$). Goodin, ¶ 189.

[1] All citations in § V are to Ex. 1007 unless otherwise indicated.

In standard floating-point representations of "normal" numbers (anything not smaller than 2^{-126}), the mantissa is kept in the range $1 \leq mantissa < 2$, so only the fraction by which the mantissa exceeds 1 is specified (*i.e.*, the mantissa's leading 1 is "implied"). Dockser, [0002]; Dockser-Lall (Ex. 1010), [0003]-[0004]; Goodin, ¶ 190. For example, if three bits are used, the mantissa 1.5 is represented with three bits (100) that represent the fraction .5 as $(1 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) = .5$, and the mantissa 1.25 is represented as the bit sequence 010 that represents the fraction .25 as $(0 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) = .25$. Goodin, ¶ 191.

It was a well-known principle of mathematics and computing that "precision… is defined by the number of bits used to represent the mantissa"—"[t]he more [mantissa] bits… the greater the precision." [0002]; Gaffar (Ex. 1012), 4; Hekstra (Ex. 1013), 4:47-52. For example, the floating-point representation of the number 29 is positive sign, exponent=4, mantissa=1.8125 (*i.e.*, $1.8125 \times 2^4 = 29$). The 1.8125 mantissa can be represented precisely as the 4-bit sequence 1101, which represents the fraction .8125 as $(1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$. If only three bits are used, the closest representable mantissas are 1.75 and 1.875—*i.e.*, the bit sequence 110 represents the mantissa $1 + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) = 1.75$, while the bit sequence 111 represents the mantissa $1 + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) = 1.875$. Thus, with a 3-bit mantissa, the number 29 cannot be represented precisely, but can be approximated less precisely as $1.75 \times 2^4 = 28$ or $1.875 \times 2^4 = 30$. Goodin, ¶¶ 192-193.

"[M]odern computers" "commonly" use the "ANSI/IEEE-754 standard… 32-bit single format," which represents floating-point numbers using "a 1-bit sign, an 8-bit exponent, and a 23-bit mantissa." [0002]. Dockser recognized that "[w]hile some applications may require these types of precision, other applications may not." [0003]. For example, if the lower precision from using just 16 mantissa bits suffices for a particular application, then "any accuracy beyond 16 bits of precision tends to result in unnecessary power consumption," which "is of particular concern in battery operated devices where power comes at a premium." [0003]; Goodin, ¶ 194.

Dockser's "floating-point processor (FPP)" operates at a selectable "subprecision of the floating-point format," below the precision of the "IEEE-754 32-bit single format" of the input numbers it receives, by removing power from some least-significant mantissa bits in the FPP's operation, resulting in those mantissa bits being dropped and "reducing the [FPP's] power consumption." [0014]-[0018], [0026]-[0027]; Goodin, ¶ 195.

As detailed below, certain implementation details (*e.g.*, the range of numbers the 32-bit IEEE-754 standard represents) were so well known as to be unnecessary for Dockser to explain. Goodin, ¶ 196; '156 patent, 2:35-37. Ground 1 establishes a POSA would have understood Dockser to teach a device implemented in these ways, or alternatively they would have been obvious implementations of Dockser's teachings. References below to Dockser "meeting" claim limitations refer to Dockser's device implemented to include any details discussed below as being the understood and/or obvious implementation of what Dockser describes.

B. Claim 1

1. [1A1] "A device comprising: at least one… (LPHDR) execution unit"

Dockser's FPP is "incorporate[d]" in a "computing system[]…, as part of the main processor or as a coprocessor." [0001], [0015], [0035]-[0036]. The computing system is a claimed "device" comprising the FPP. Goodin, ¶¶ 197-198; '156 patent, 1:64, 7:45-48, 23:66-67, 27:52-55, 28:61-29:4 (claimed "device" includes "a computer including a processor and other components").

Dockser's FPP is an "execution unit" as claimed. Goodin, ¶ 199. It is a "specialized computing unit[] that perform[s] certain mathematical operations, e.g., multiplication, [etc.]" by "execut[ing]… instructions" including "multiply instructions." [0001], [0019], [0024]; Goodin, ¶ 199. Like the "processing element" (PE) the '156 patent describes as one possible embodiment of an LPHDR "execution unit" ('156 patent, 8:12-16, 8:30-33), Dockser's FPP is a "unit[] which pairs memory with arithmetic" and implements the memory as registers ('156 patent, 16:56-58; FIG. 4); both Dockser's FPP and the '156 patent's PE include registers and an arithmetic unit to perform arithmetic operations on data locally stored in the registers. ’156 patent, FIGs. 4, 6, 10:38-11:43, 12:53-13:3; Dockser, FIG. 1, [0015]-[0025]; Goodin, ¶¶ 200-207.

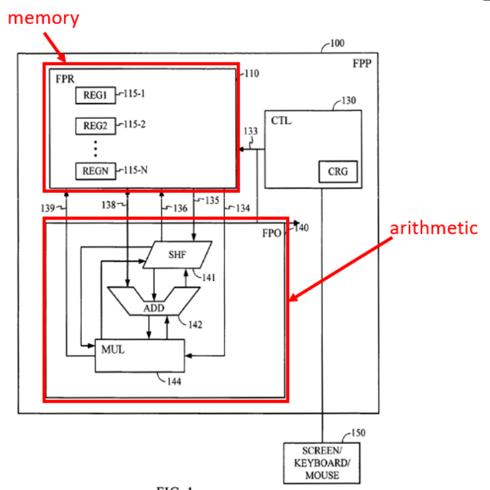


FIG. 1

Dockser, FIG. 1 (annotated)

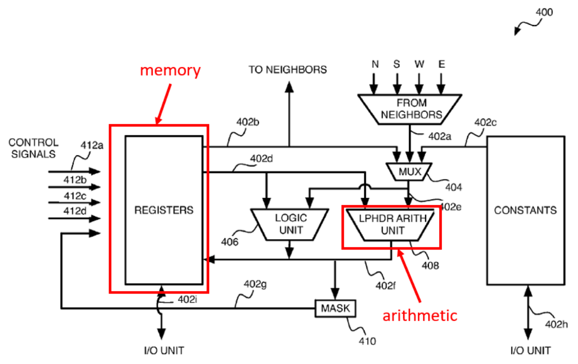


FIG. 4

’156 Patent, FIG. 4 (annotated)

In an alternative mapping, the floating-point operator (FPO) inside Dockser’s FPP is also a claimed “execution unit” because it “include[s]… components configured to perform… floating-point operations,” including a floating-point multiplier (MUL) that “execute[s] floating-point multiply instructions.” [0019]; Goodin, ¶ 208. References herein to “Dockser’s execution unit” meeting a claim limitation refer to both Dockser’s FPP and FPO meeting the limitation in the same way because the FPP includes the FPO.

Dockser’s FPP (including its FPO) is “low precision” as claimed because “the precision” of operations in the FPP is “reduced” (*e.g.*, [0014]), and because it operates with the minimum imprecision in [1B2]. Goodin, ¶¶ 209-210; *infra* § V.B.4; ’156 patent, 26:39-27:51 (characterizing “degrees of precision” described in language mirroring that in [1B2] as “low precision”); Ex. 1035, 8 (Singular’s litigation assertion that “low precision” and “high dynamic range” are “defined in the claim”).

The ’156 patent describes a 6-bit floating-point exponent as “provid[ing] the desired high dynamic range.” ’156 patent, 14:56-64. Dockser’s FPP (including its FPO) is “high dynamic range” because it uses an 8-bit floating-point exponent ([0017]) that provides an even higher dynamic range, and meets the range in [1B1]. Goodin, ¶ 211; *infra* § V.B.3; ’156 patent, 26:61-27:17.

2. [1A2] “adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value”

Dockser’s FPP is “adapted to execute a first operation” as claimed because it “perform[s]… mathematical ***operations***,[2] e.g., multiplication.” [0001], [0004]-[0008], [0019] (FPP components including FPO are “configured to ***execute***” instructions of “***operations***”), [0024]. Goodin, ¶ 212.

The FPP executes the operations at reduced precision on “operands” (“32-bit binary floating-point ***number[s]***” ([0016]-[0017], [0024])) that represent numerical values, as claimed. Goodin, ¶¶ 213-214.

[2] Emphasis added throughout.

The ’156 patent’s “processing element” (PE) 400 (FIG. 4) is one example embodiment of an “execution unit… for performing LPHDR operations.” ’156 patent, 8:12-33, 10:38-40, 2:15-35, 6:3-7. PE 400 receives input via an “I/O [Input/Output] Unit” and “stores” input values in “Registers” as “local data” on which the PE “operate[s]” and “perform[s] computations.” *Id.*, 9:11-21, 10:38-62, 8:42-45, FIGs. 1, 4 (annotated below); Goodin, ¶¶ 215-216.

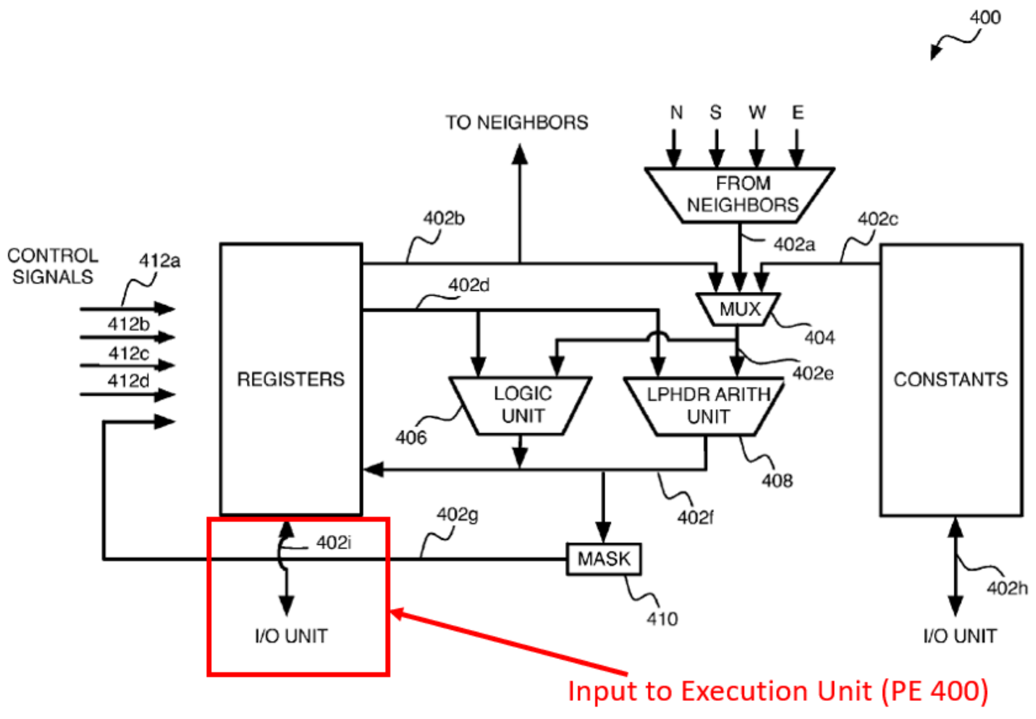


FIG. 4

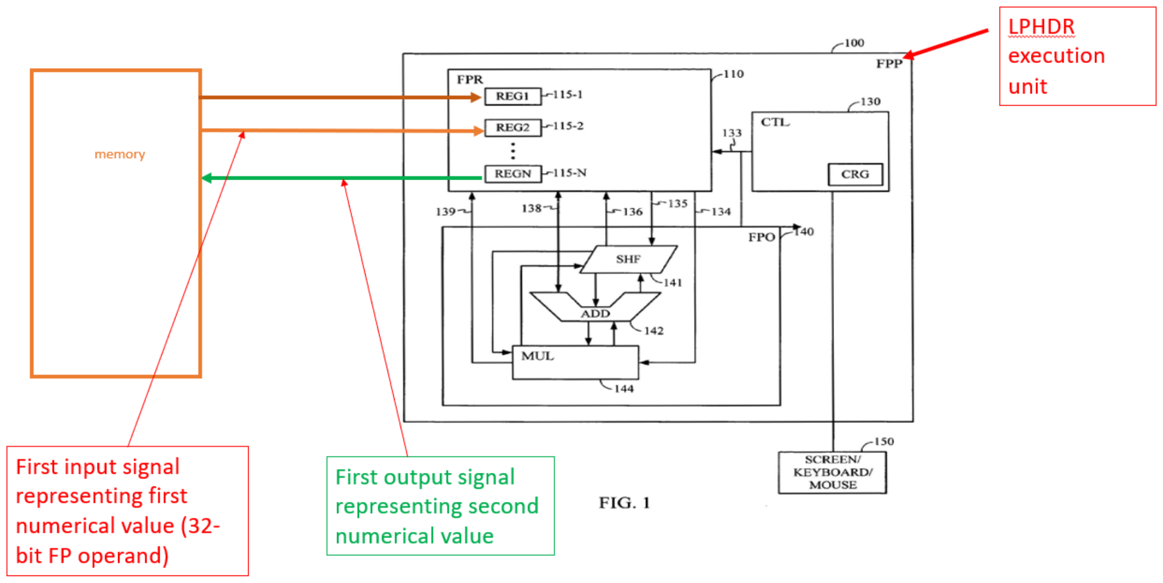
“The input… received by… PE 400 may… take the form of electrical signals representing numerical values.” ’156 patent, 11:1-4. The PE “operate[s] on its local data” (*e.g.*, input values received at the Registers) via “data paths 402*a*-*i*” through “routing mechanisms… and components” that represent values as electrical signals. *Id.*, 10:62-11:4; Goodin, ¶¶ 217-218. A POSA would have understood from the patent's description that the claimed execution unit's "first input signal representing a first numerical value" encompasses an input electrical signal (*e.g.*, 402i in FIG. 4) input to PE 400 (an example "execution unit") at a register as data on which the PE performs an operation via the PE's "data paths… and components." Goodin, ¶ 219.

Like the '156 patent's example PE embodiment, Dockser's FPP receives input values (operands) at registers, and performs operations on these inputs via the FPP's data paths 134-139 and components 140-144. FIG. 1; Goodin, ¶ 220. The operands are "move[d]" into the FPP's registers as "data from [the computer system's main] memory" in 32-bit format. [0016]-[0017]. A POSA would have understood Dockser to describe that the operand "data" being input to the FPP is a "first input signal representing a first numerical value" (*i.e.*, representing the operand "number" [0017]) as claimed, because computing circuits like Dockser's "move" data between components (*e.g.*, from memory to processor) via electrical signals. Goodin, ¶ 221; Begun (Ex. 1014), 1:36-37. Alternatively, to the extent Dockser is not considered to expressly disclose that the data is moved to the FPP via electrical signals, that would have been the straightforward and obvious way to implement what Dockser describes. Goodin, ¶ 222; Begun, 1:36-37.

Dockser's FPP is thus "adapted to execute a first operation [*e.g.*, reduced-precision multiplication] on a first input signal representing a first numerical value [operand]" as claimed. Goodin, ¶ 223. The FPP's execution of the operation produces an "***output value***" (also called "***output number***"), represented by "output bits" ([0034]), that is a "second numerical value" as claimed. Goodin, ¶¶ 224-227.

The output value is sent to a register ([0024]), from where it is "move[d]… to… the main memory" ([0016]). Goodin, ¶ 228. A POSA would have understood Dockser to describe that the output value sent as bits from the FPP's registers to main memory is a "first output ***signal*** representing a second numerical value" as claimed, because circuits like Dockser's "move" data between such components via electrical signals. Goodin, ¶ 229; Begun, 1:36-37. Alternatively, to the extent Dockser is not considered to expressly disclose that the data is moved from the FPP to memory via electrical signals, that would have been the straightforward and obvious way to implement what Dockser describes. Goodin, ¶ 229; Begun, 1:36-37.

Thus, Dockser's FPP is "adapted to execute [the] first operation [*e.g.*, reduced-precision multiplication]… to produce a first output signal [electrical signal sending output to memory] representing a second numerical value [the output number from the multiplication]" as claimed. Goodin, ¶ 230.



Dockser, FIG. 1 (annotated)

In the alternative mapping where Dockser's FPO meets the claimed "execution unit" (*supra* § V.B.1), the FPO is "adapted to execute [the] first operation [reduced-precision multiplication] on a first input signal representing a first numerical value [signal 134 inputting operands to FPO] to produce a first output signal representing a second numerical value [signal 139 outputting multiplication result from FPO]" as claimed. Goodin, ¶ 231; *infra* § V.B.4.c(2).

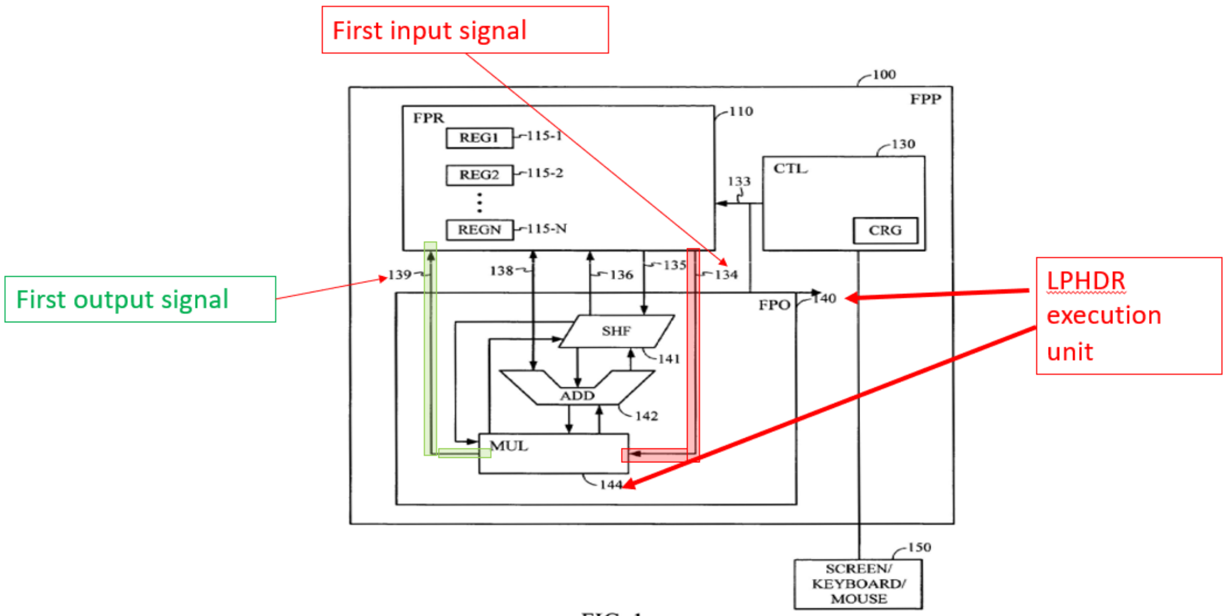


FIG. 1

Dockser Figure 1 (annotated)

3. [1B1] "wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000"

Dockser's FPP operates on IEEE-754 32-bit single-format numbers having 8-bit exponents. [0002], [0016]-[0017], [0024]; *supra* § V.B.1; Goodin, ¶ 232. The number of exponent bits determines dynamic range. Goodin, ¶ 232; Gaffar, 4; Hekstra, 4:47-52; Grinberg (Ex. 1015), 8:48-49.

IEEE-754 8-bit exponents range from -126 to 127; thus the dynamic range of "normal" IEEE-754 single-format operands is from around 2^{-126} (much smaller than 1/65,000) to around 2^{127} (much larger than 65,000). Goodin, ¶ 233; Dockser-Lall, [0004]; Tong, 274. A POSA would have understood Dockser to disclose that its FPP supports these "normal" IEEE-754 single-format operands, as Dockser makes no reference to supporting far less common "denormal" numbers (smaller than 2^{-126}), let alone to supporting denormal numbers exclusively. Goodin, ¶ 234; Dockser-Lall, [0005], [0008]. To the extent Dockser is not considered to expressly disclose that the FPP operates on "normal" IEEE-754 single-format operands, that would have been a conventional and obvious way to implement what Dockser discloses. Goodin, ¶ 234. Thus, "the dynamic range of the possible valid inputs" (normal IEEE-754 single-format operands) to Dockser's FPP operation "is at least as wide as from 1/65,000 through 65,000" as claimed. Goodin, ¶ 235.

As discussed below (§ V.B.4.c), Dockser's precision-reduction techniques within the FPP do not change the exponents of operated-on numbers; thus the dynamic range of the possible valid inputs in the alternative mapping where the FPO meets the claimed "execution unit" (*supra* § V.B.1) is the same as that of the FPP inputs. Goodin, ¶ 236.

4. [1B2] "for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input"

a. "Statistical Mean" Limitation

The specification's only mention of "the statistical mean, over repeated execution of… [the] operation on each… respective input[]" ('156 patent, 27:33-49) references "non-deterministic" embodiments (*id*., 27:20-33). Goodin, ¶ 237. A POSA would have understood the claims' "statistical mean" limitation in the context of the '156 patent's stated intent to claim not only "repeatable" deterministic embodiments like digital circuits that always produce the same output when repeating an operation on the same input, but also analog embodiments that are non-deterministic because they "introduce noise into their computations, so the computations are not repeatable." '156 patent, 4:11-17, 14:19-64, 17:12-18; Goodin, ¶ 238.

For example, if an analog circuit represents the number 1 as a voltage somewhere between 0.99 and 1.01 ('156 patent, 14:15-33), a first execution of an operation that adds 1+1 may produce a result of 1.98, a second execution may produce 2.01, and repeated executions may produce different results unpredictably. Goodin, ¶ 239. POSAs understood that for such non-deterministic embodiments, the claimed "statistical mean" limitation refers to averaging the different outputs produced by the same operation on the same input—*e.g.*, the mean of 1.98 and 2.01 is 1.995. Goodin, ¶ 240.

For deterministic digital embodiments, the claimed "statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input" is the same as the numerical value of the first output signal for any individual execution of the first operation on each specific input, because that output is always the same for any specific input. Goodin, ¶ 241; Ex. 1032, ¶ 129 (Singular's same assertion in litigation).

Dockser uses "a conventional floating-point multiplier" ([0020]), which a POSA would have understood is a conventional deterministic digital circuit (Goodin, ¶ 242; Weiss (Ex. 1011), 1:40-42), so repeatedly executing Dockser's multiplication operation on the same input (*i.e.,* pair of operands) with the same precision level yields the same result for every execution; therefore, the statistical mean of the outputs is the same as the output for any single execution. Goodin, ¶ 243. To the extent Dockser is not considered to expressly disclose that its "conventional floating-point multiplier" ([0020]) is a deterministic digital circuit, that would have been the conventional and obvious implementation. Goodin, ¶ 244; Weiss, 1:40-42; Ex. 1022, 5:14-21 ('156 patent's inventor explaining that "digital silicon" circuits are "deterministic, which programmers like").

b. “Exact Mathematical Calculation” Limitation

The numerical value of each input operand to Dockser’s operation (reduced-precision multiplication) is the number represented by the IEEE-754 32-bit sequence ([0016]-[0018]; *supra* §§ V.B.2-V.B.3), and the claimed “result of an exact mathematical calculation of the first operation on the numerical values of that same input” is the (>32-bit) product that would result if the pair of input 32-bit operands were multiplied ***without*** reducing precision. Goodin, ¶ 245; *see infra* Appendix I.B.[3] As discussed *infra* § V.B.4.c, the numerical value represented by Dockser’s output signal differs from this “exact mathematical calculation” because Dockser performs reduced-precision multiplication. Goodin, ¶ 245.

c. Relative Error (Y) for Fraction of Valid Inputs (X) Limitation

[1B2] recites that “for at least X=5% of the possible valid inputs to the first operation,” the statistical mean limitation (met in Dockser by the numerical value represented by the output signal for any single execution of the first operation on a specific valid input—*supra* § V.B.4.a) “differs by at least Y=0.05% from” the exact mathematical calculation discussed *supra* § V.B.4.b. Goodin, ¶ 246. The ’156 patent refers to the Y percentage as the “relative error amount E by which the result calculated by [the] LPHDR element… differ[s] from the mathematically

[3] The Petition’s word count includes Appendix I.

correct result," and the X percentage as the "fraction F of the valid inputs" for which the operation produces at least that relative error amount. '156 patent, 27:29-51; Goodin, ¶ 247.

The claimed "possible valid inputs" are met in Dockser by the set of possible normal IEEE-754 32-bit single-format numbers forming pairs of operands in input signals to the execution unit that can be multiplied together to produce an output representing a numerical value (rather than, *e.g.*, an overflow/underflow exception). [0016]-[0017], [0024]-[0027]; *supra* § V.B.3; Goodin, ¶ 248. As discussed below, Dockser meets [1B2] because Dockser operates the FPP and FPO at precision levels where the claimed relative error (Y) is at least 0.05% for at least X=5% of all valid input pairs of normal IEEE-754 32-bit single-format numbers. Goodin, ¶ 249.

The precision level is selected by specifying the number of mantissa fraction bits to be retained; the remaining "excess bits" are dropped. [0004]-[0007], [0017], [0026]-[0029]; FIG. 2; Goodin, ¶¶ 250-254. In one "example," Dockser retains only the 9 most-significant bits (MSBs)—*i.e.*, the leftmost 9 bits—of the mantissa fraction, and drops the remaining 14 mantissa bits. [0025]-[0028]; Goodin, ¶ 255.

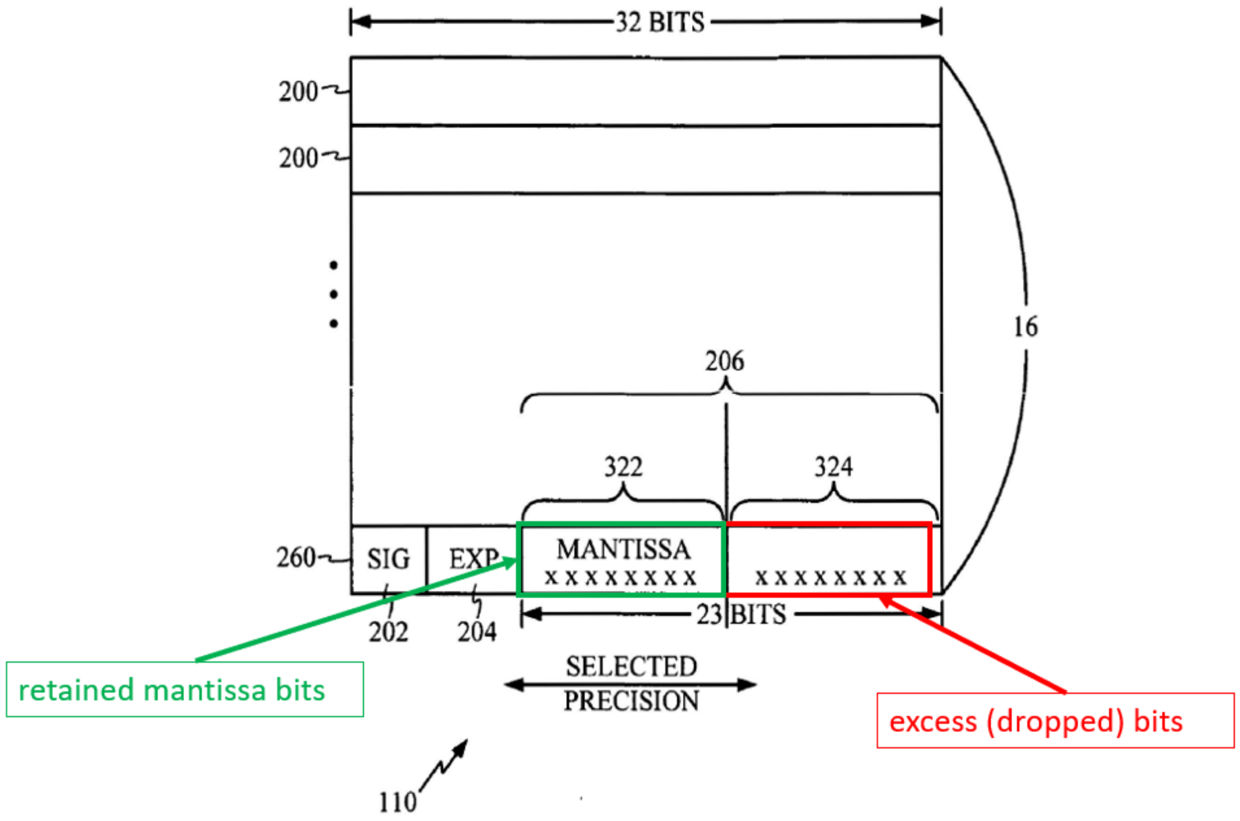


FIG. 2

Dockser, Figure 2 (annotated)

Dockser discloses two precision-reducing techniques that can be used separately or together. [0004]-[0007], Dockser claims 9-11; Goodin, ¶¶ 256-262. One technique drops bits from the operands by removing power from storage elements in the FPP's registers that correspond to the excess (dropped) mantissa bits. [0006], [0026]; Goodin, ¶ 257. The other technique drops bits by removing power from elements within the multiplier logic that computes the product of the operand mantissas. [0007], [0027], [0030]-[0034]; Goodin, ¶ 258.

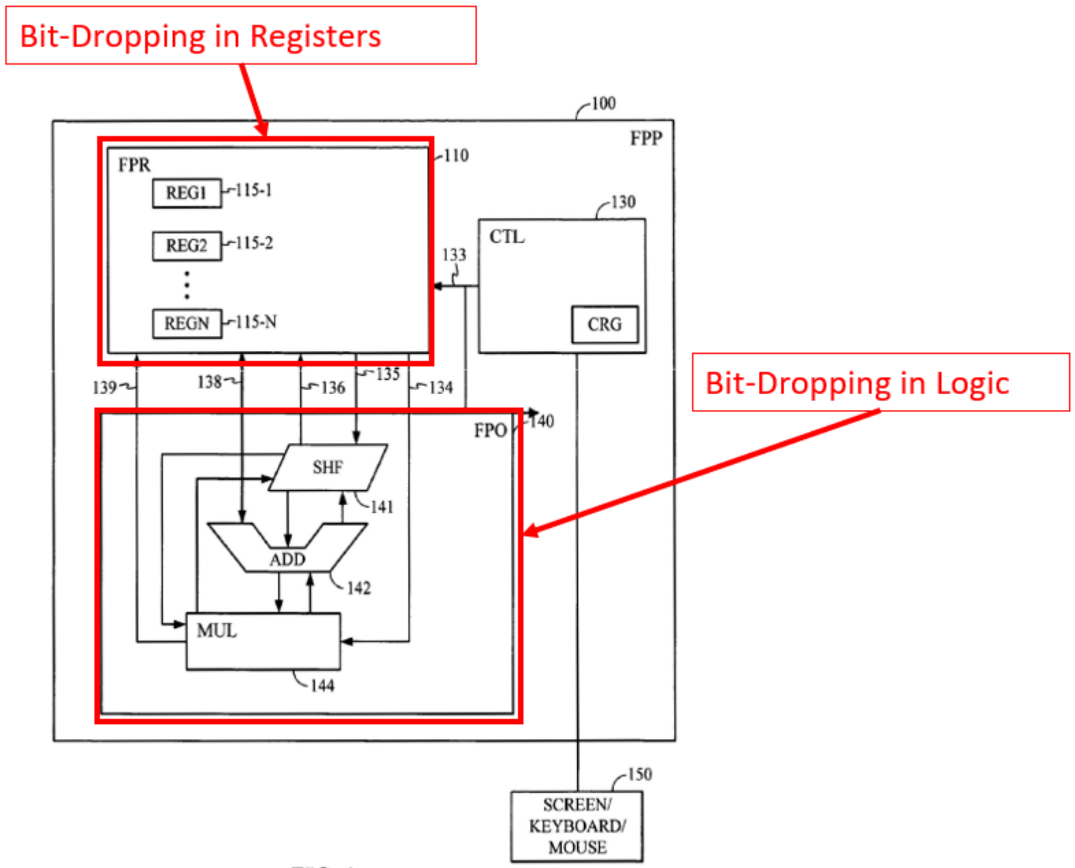


FIG. 1

Dockser, Figure 1 (annotated)

(1) Dockser's Register Bit-Dropping Meets [1B2]

A POSA would have understood Dockser's "conventional floating-point multiplier" ([0020]) multiplies two floating-point operands by (in relevant part) adding their exponents and multiplying their mantissas, as the '156 patent (at 14:65-15:1.) acknowledges was the "traditional floating point method[]." Goodin, ¶ 263; Tong, 274-275. This conventional technique applies basic properties of multiplication and exponents where $(M_1 \times 2^{E_1}) \times (M_2 \times 2^{E_2}) = (M_1 \times M_2) \times 2^{E_1+E_2}$. Goodin, ¶ 263. Because Dockser reduces precision by dropping mantissa bits ([0002]-[0003], [0026]), a POSA would have understood the mantissa multiplication produces error relative to an "exact mathematical calculation." Goodin, ¶ 264.

Specifically, Dockser's FPP receives operands into registers 115 as IEEE-754 single-format floating-point numbers with 23 bits representing the mantissa's fractional part (as discussed *supra* § V.A, the "1" in the mantissa's integer part is implied), but the register bit-dropping technique removes power from the register storage elements for "excess" least-significant mantissa bits to the right of the selected precision level in FIG. 2. [0006], [0016]-[0018], [0026]-[0028], FIGs. 1-2 (reproduced below, annotated); Goodin, ¶¶ 265-268.

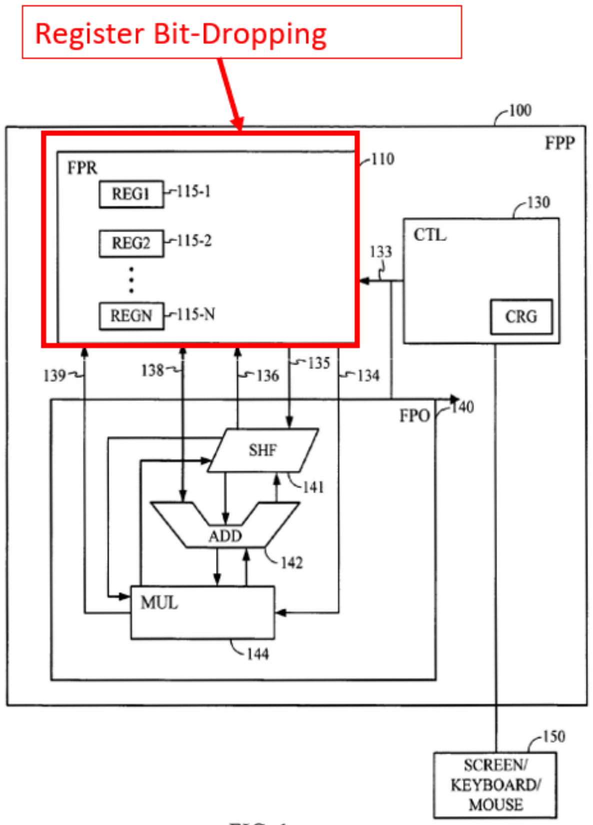


FIG. 1

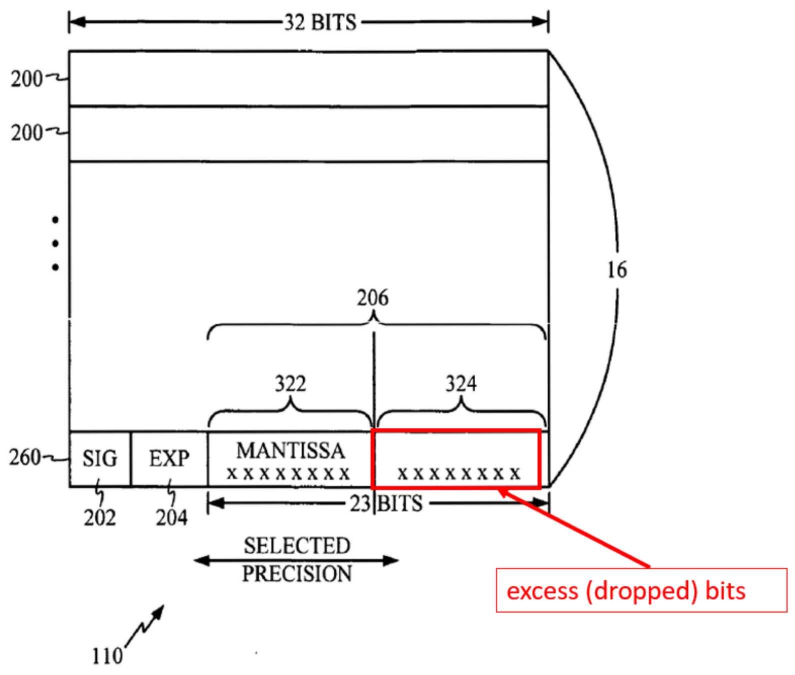


FIG. 2

"By way of example," if a precision level retaining 9 mantissa fraction bits is selected, "[p]ower can be removed from the floating-point register elements for the remaining 14 [mantissa] fraction bits." [0026]; Goodin, ¶ 269. A POSA would have understood the output of unpowered storage elements would be tied to zero voltage (*e.g.*, ground), making those 14 "excess" bits zeroes. [0026], [0029]; Goodin, ¶¶ 270-272 (citing Hawkins (Ex. 1020), 1:13-2:15 (unpowered elements should be tied to ground to prevent them from "draw[ing] unacceptably large amounts of power" and "shorting… the power supply"); Youngs (Ex. 1060), [0005]; Flynn (Ex. 1019), [0003]; Cohen (Ex. 1016), 3:63-66, 4:51-55 (interpreting low voltage as "0" is "typical[]…[in] digital computer system[s]")). To the extent

Dockser is not considered to disclose that the outputs of the unpowered register storage elements would be tied to ground to represent "0," that would have been the straightforward, well-known, conventional, and obvious way to implement Dockser's described unpowering of register storage elements, as corroborated, *inter alia*, by Hawkins, Youngs, Flynn and Cohen. Goodin, ¶ 273.

As representative examples, the 23-bit sequence 01100000111100111001110 (representing a mantissa of 1.37871718406677246609375) would become 01100000100000000000000 (representing a mantissa of 1.376953125) by zeroing the 14 right-most bits, and 01000100110111000001000 (representing a mantissa of 1.26898288726806640625) would become 01000100100000000000000 (representing a mantissa of 1.267578125). Goodin, ¶ 274.

A pair of operands with "excess" bits dropped from the registers are multiplied together, producing a reduced-precision output that differs from the "exact" product of the original 32-bit operands. [0024]-[0026]; Goodin, ¶ 275. For instance, if the two example 23-bit-fraction mantissas above were multiplied, the exact mantissa product would be 1.7495685129631510790204629302025; whereas the reduced-precision mantissa product output from Dockser's register bit-dropping technique would be 1.745395660400390625 (resulting from multiplying the mantissas represented by the fraction bit sequences 01100000100000000000000 and 01000100100000000000000). Goodin, ¶ 276.

Because Dockser's output mantissa differs from the exact mantissa product, the numerical value represented by the output floating-point number (composed of mantissa, exponent, and sign) differs from the exact product of the original floating-point operands. Goodin, ¶ 277. A POSA would have understood, by straightforward math detailed in Appendix I.A *infra*, that the relative error (the claimed "Y" percentage) of any floating-point number output from Dockser's reduced-precision multiplication is the same as the relative error of its mantissa, independent of its exponent and sign. Goodin, ¶ 278.

A POSA would thus have understood Dockser's register bit-dropping technique produces relative error, the amount of which depends on the number of mantissa bits dropped—the more bits dropped, the greater the error. Goodin, ¶ 279.

Claim 1 characterizes the claimed execution unit using ***performance*** characteristics of a minimum relative error (Y) and a minimum fraction (X) of the possible valid inputs that meet Y. Claim 1 recites no ***structural*** characteristics of the execution unit, and the specification provides no guidance how to evaluate any execution unit's structure to determine whether it meets the claimed performance characteristics. Goodin, ¶ 280.

A POSA asked whether Dockser's FPP meets the claimed imprecision performance characteristics would have understood that Dockser suggests any desired number of mantissa bits can be dropped, and that numerous obvious implementations of Dockser's FPP drop sufficient bits to yield an execution unit meeting the claimed minimum imprecision. Goodin, ¶ 281. If asked to quantify how many bits dropped would meet the claimed X and Y, a POSA would have taken one of two approaches.

(i) Software Demonstration

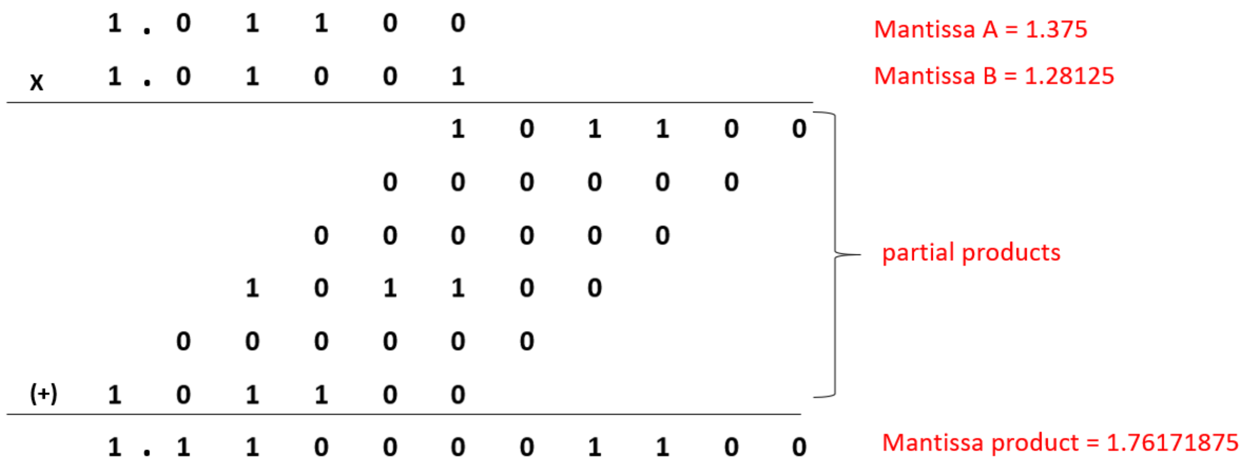
Given the massive number of possible inputs to Dockser's FPP (including over 70 trillion possible pairs of normal IEEE-754 single-format mantissas), a POSA would have performed Dockser's FPP operation in software to determine the fraction X of all possible valid inputs that produce at least the claimed relative error Y when a given number of mantissa bits are dropped. Goodin, ¶ 282. As detailed in Appendix I.B *infra*, Mr. Goodin did just that. Specifically, he wrote a software program (which a POSA would have understood how to write) that performs floating-point multiplication operations the way Dockser's register bit-dropping technique does when dropping the 14 least-significant mantissa bits to zero as taught by Dockser. Dockser, [0026]; Goodin, ¶ 283. The program tested all possible valid pairs of normal IEEE-754 single-format operands (by testing all possible valid mantissa pairs, because the relative error Y is independent of exponent and sign—*see infra* Appendix I.A), and demonstrates that Dockser's register bit-dropping technique, when performed with a selected precision level retaining 9 mantissa fraction bits as Dockser ([0026]) discloses, produces at least Y=0.05% relative error for 92.1% of possible valid inputs (greater than X=5%), meeting [1B2]. Goodin, ¶ 284.

(ii) Pencil-and-Paper Algebraic Demonstration

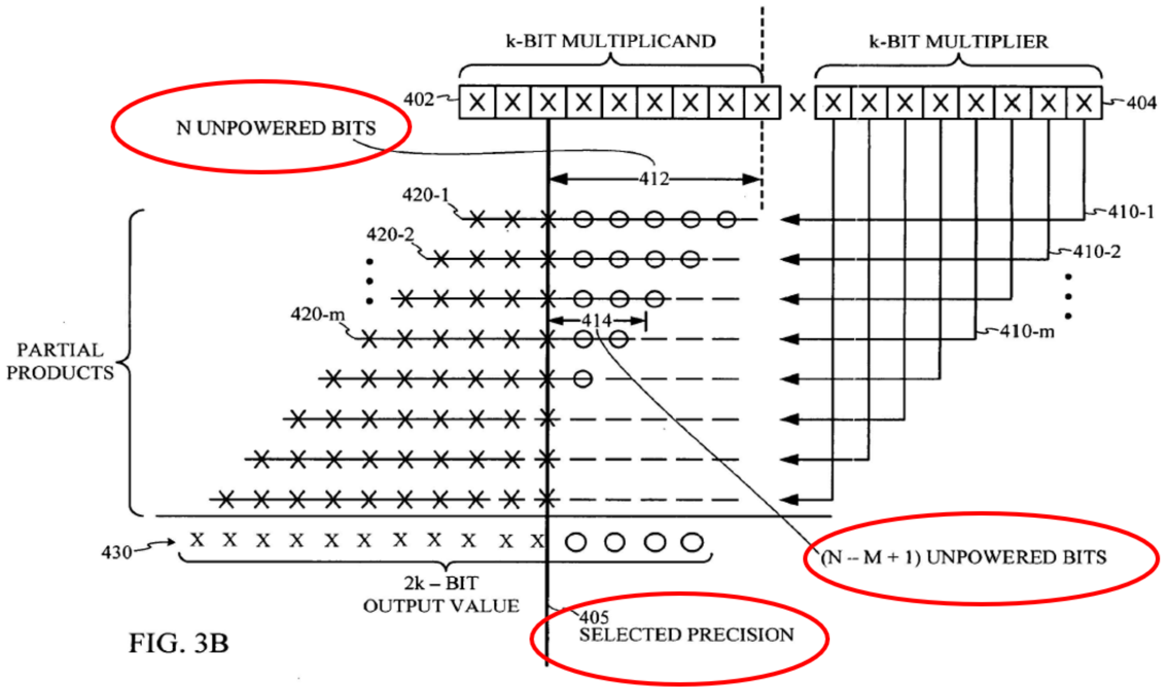
A POSA would also have understood algebraically that Dockser's register bit-dropping technique meets [1B2], by examining the absolute ***minimum*** relative error produced by zeroing certain mantissa bit positions. Goodin, ¶ 285. As detailed in Appendix I.C *infra*, over 12% of all possible valid normal IEEE-754 single-format operands have a zero as their most-significant (leftmost) mantissa fraction bit and ones as their tenth and eleventh fraction bits. When Dockser's register bit-dropping retains only nine fraction bits, ***every*** input in that 12% produces at ***minimum*** 0.097% relative error, which meets [1B2]'s X and Y. Goodin, ¶ 286.

(2) Dockser's Multiplier Logic Bit-Dropping Meets [1B2]

Dockser's second precision-reduction technique removes power from bits within the FPO's logic that multiplies the operand mantissas. [0007], [0027], [0030]-[0034]; Goodin, ¶ 287. This logic operates conventionally, computing the mantissa product by generating and adding together "partial products" similar to pencil-and-paper long multiplication. [0030]-[0031]; Goodin, ¶¶ 288-290. This is illustrated in the simple example below using 5-bit mantissa fractions. Each partial product results from multiplying the multiplicand (Mantissa A) by the "0" or "1" at each position in the multiplier (Mantissa B).

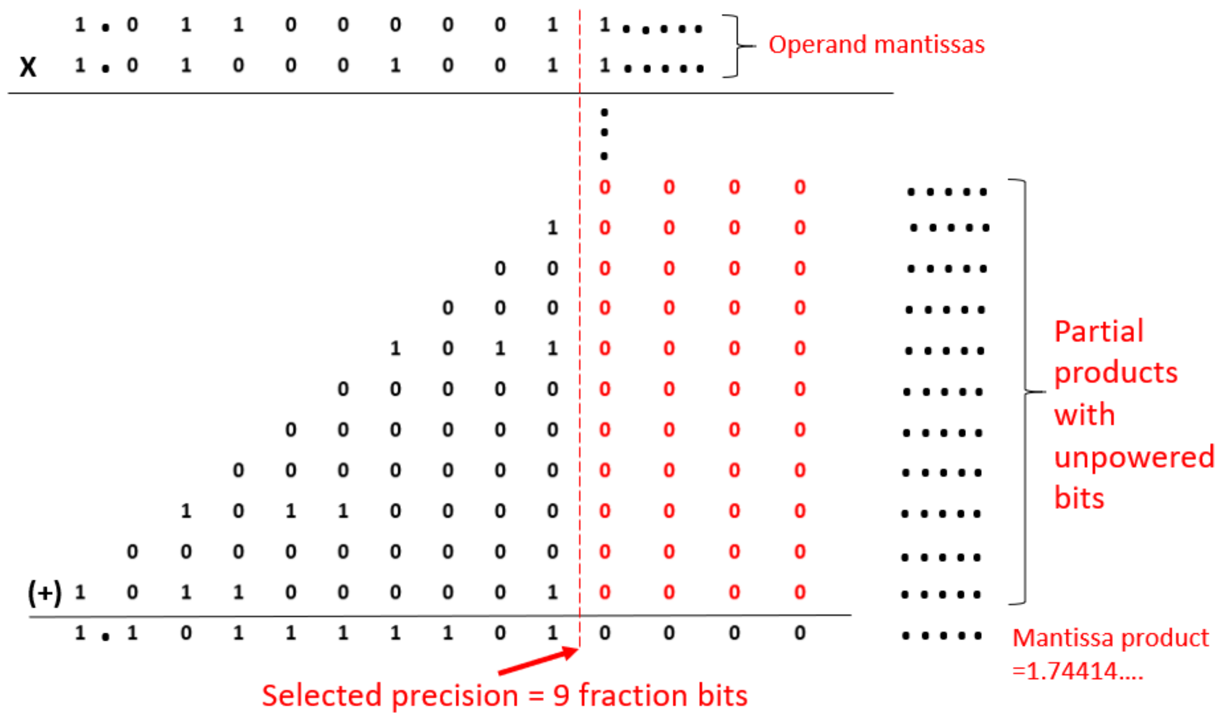


Power is removed from the logic that implements the partial product bits to the right of line 405 corresponding to a precision level selected for output value 430. Dockser, [0032]-[0034]; Goodin, ¶¶ 291-292. A POSA would have understood Dockser to disclose that the bits from which power is removed in the logic are dropped to 0s, or alternatively this would have been the conventional and obvious way to implement the power removal Dockser describes. Goodin, ¶ 293; *see supra* § V.B.4.c(1).



Dockser, Fig. 3B (annotated)

An example using the same exemplary operand mantissas discussed *supra* § V.B.4.c(1) is illustrated below, in which dropping partial-product bits to the right of the selected precision produces an output mantissa product of 1.744140625, which differs from the exact product of 1.7495685129631510790204629302025. Goodin, ¶ 294.


Operand mantissas
Partial products with unpowered bits
Mantissa product =1.74414....
Selected precision = 9 fraction bits

A POSA would have understood the relative error introduced depends on the number of mantissa bits dropped—the more bits dropped, the greater the error. Goodin, ¶ 295.

If asked whether Dockser's FPP using a selected precision level retaining 9 bits (as Dockser [0026] teaches) in the output mantissa by dropping less-significant bits in the multiplier logic meets the claimed X and Y, given the massive number of possible inputs Dockser's FPP supports (*supra* § V.B.4.c(1)(i)), a POSA would have written a software program. Goodin, ¶ 296. As detailed in Appendix I.D *infra*, Mr. Goodin wrote a software program (which a POSA would have understood how to write) that performs floating-point multiplication operations the way Dockser's logic bit-dropping technique does by dropping to zero the mantissa partial product bits less significant than the 9-bit selected precision level. Goodin, ¶ 297. Mr. Goodin's program tested all possible valid pairs of normal IEEE-754 single-format operands (by testing all possible valid mantissa pairs—*supra* § V.B.4.c(1)(i); *infra* Appendix I.A, I.D) and demonstrates that Dockser's multiplier logic bit-dropping technique, when performed with a precision level retaining 9 output mantissa fraction bits, produces at least Y=0.05% relative error for 99.35% of possible valid inputs (greater than X=5%), meeting [1B2]. Goodin, ¶ 297.

In the alternative mapping (*supra* §§ V.B.1-V.B.2), the operand numbers input via 134 (claimed "first input signal") to the FPO (claimed "execution unit") are the same as those input to the FPP registers when Dockser's logic bit-dropping technique is used alone, and the same output numbers from the FPO are output from the FPP. Goodin, ¶ 298. Therefore Mr. Goodin's program demonstrates that Dockser meets [1B2] under either mapping. Goodin, ¶ 298.

(3) Register and Multiplier Logic Bit-Dropping Together Also Meets [1B2]

In an implementation of Dockser's FPP that drops bits in ***both*** the registers and the multiplier logic, where Dockser's FPP is the claimed "execution unit," the amount of relative error is the same as produced by the logic bit-dropping alone, if the same selected precision level (retaining the same number of mantissa fraction bits) is applied at the registers and in the logic. Goodin, ¶ 299. This is because at least the same bits that are dropped to zero in the register bit-dropping are dropped to zero in the corresponding partial products in the logic bit-dropping, for the same selected precision level. Goodin, ¶¶ 299-301. Thus, the results of the same software program that demonstrate that Dockser's multiplier logic bit-dropping alone meets [1B2] also demonstrate that Dockser's register and logic bit-dropping, performed together at a selected precision level retaining 9 mantissa fraction bits as Dockser ([0026]) discloses, meet [1B2]. Goodin, ¶ 302; *supra* § V.B.4.c(2).

5. [1C] "at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit"

Dockser's computing system includes a "main processor," meeting the claimed "computing device." [0015], [0035]; '156 patent, 7:45-60 ("computing device[]" implementable as "processor"); Goodin, ¶ 303. Dockser's main processor meets [1C] because it writes "subprecision select bits" to the FPP's "***control*** register," and thus is adapted to control the operation of the FPP (and its FPO) by specifying its precision level. [0018], [0025]; Goodin, ¶ 304.

C. Claim 2

Claim 2 lists alternatives for implementing the "computing device," only one of which must be met to render claim 2 unpatentable. *Apple v. Evolved Wireless*, IPR2016-01177, Paper 27 at 9-10 (Dec. 20, 2017); *Daifuku Co. v. Murata Mach.*, IPR2015-00083, Paper 63 at 13-14 (May 3, 2016); ’156 patent, 25:30-41, 28:42-60 (discussing claim 2’s recited alternatives disjunctively).

Claim 2 is met because Dockser discloses that its “main processor” (the “computing device” of [1C] and claim 2) is a “general-purpose processor” implementable as a “state machine” as claimed. [0035] (FPP is “part of a general purpose processor” that “may be any conventional processor,… or state machine”), [0015] (FPP is “part of the main processor”); Goodin, ¶ 305.

Claim 2 is alternatively met because a “CPU” was a well-known “conventional processor” of the type Dockser discloses ([0035]), and it would have been obvious to implement Dockser’s main processor as a CPU as claimed. Goodin, ¶ 306; Ex. 1023, 1:16-28.

D. Claim 16

Independent claim 16 recites a device comprising a “plurality of components” comprising at least one LPHDR execution unit that is exactly as recited in claim 1.

1. [16A1] “A device comprising: a plurality of components comprising: at least one first… (LPHDR) execution unit”

Dockser’s computing system device meets [16A1] because it comprises a plurality of components including a main processor and an FPP (with FPO) (a claimed “LPHDR execution unit”). Goodin, ¶ 307; *see supra* §§ V.B.1, V.B.5.

If claim 16 were alternatively interpreted to require that a plurality of components be included in a single LPHDR execution unit, Dockser's FPP has multiple components (*e.g.*, registers, controller, FPO), as does Dockser's FPO (adder, multiplier). [0015], [0019]-[0022]; Goodin, ¶ 308.

2. [16A2]-[16B2]

[16A2]-[16B2] are identical to [1A2]-[1B2] and met for the same reasons discussed *supra* §§ V.B.2-V.B.4. Goodin, ¶¶ 309-310.

VI. <u>GROUND 2</u>: CLAIMS 1-2, 16, AND 33 WOULD HAVE BEEN OBVIOUS OVER DOCKSER AND TONG

A. Tong

Tong (Ex. 1008) is Section 102(b) prior art because it was publicly accessible by June 30, 2000. Tong's face bears a 2000 copyright date and indicates it was published by the IEEE, a "well-known, reputable compiler and publisher of scientific and technical publications," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 3, June 2000. *Ericsson v. Intellectual Ventures I*, IPR2014-00527, Paper 41 at 10–11 (May 18, 2015) (finding IEEE publication copyright line information falls under FRE 803(17) hearsay exception); *see also Hulu v. Sound View Innovations*, IPR2018-01039, Paper 29 at 19 (Dec. 20, 2019) (precedential) (finding reasonable likelihood of printed publication based on copyright date, printing date, and ISBN date from "established publisher" of "well-known book series"); Goodin, ¶ 311.

Additionally, an IEEE representative (Mr. Grenier) testified that the IEEE made Tong publicly available by June 30, 2000. Ex. 1025, ¶¶ 8-11. *Guest Tek Interactive Entm't v. Nomadix*, IPR2019-00253, Paper 6 at 27-31 (May 28, 2019) (crediting IEEE copyright/publication date indicia and Mr. Grenier's testimony of public availability); *Palo Alto Networks v. Finjan*, IPR2015-01974, Paper 49 at 19-20 (Mar. 16, 2017) (same). Other IEEE publications cited Tong before 2009, further corroborating Tong's public accessibility. Ex. 1026, 27 (citation [13]); Ex. 1027, 27 (citation [31]); *see Intel v. R2 Semiconductor*, IPR2017-00707, Paper 81 at 21-22 (July 31, 2018).

Tong, like Dockser (*supra* § V.B.4.c; *infra* § VII.E), confirms that the number of mantissa bits used in a high-dynamic-range floating-point execution unit was a well-known result-effective variable impacting power consumption and precision. Tong, 278 ("[P]ower dissipation in an FP [floating-point] unit can be reduced by using fewer bits in the FP representation. However, fewer bits reduces precision"); Goodin, ¶¶ 312-313. Focusing on "signal processing applications" "long… known" to be able to "get by with less precision/range than full FP" (Tong, 273), Tong explains that "[s]ince multiplication is one of the most frequent operations in signal processing applications" (Tong, 274) and "the mantissa multiplier always consumes more power than all other blocks" in an FP multiplier

(Tong, 275-276), "reduction in the mantissa bitwidth is the most effective means of reducing power dissipation" (Tong, 277). Goodin, ¶ 314.

Tong teaches how to determine, by "emulat[ing] in software different bitwidth FP units" and "plot[ting] the accuracy" of various signal-processing applications "across a range of mantissa bitwidths" (Tong, 278), "the *minimal* number of mantissa… bits" for each application "to *reduce* power consumption, yet *maintain* [the] program's overall accuracy" (Tong, 273, emphasis original). Goodin, ¶¶ 315-319. In the "Sphinx" speech recognition application and the "ALVINN" navigation application, "the accuracy does not change significantly with as few as 5 mantissa [fraction] bits." Tong, 278-279, 273, 282, Fig. 6; Goodin, ¶¶ 315-319.

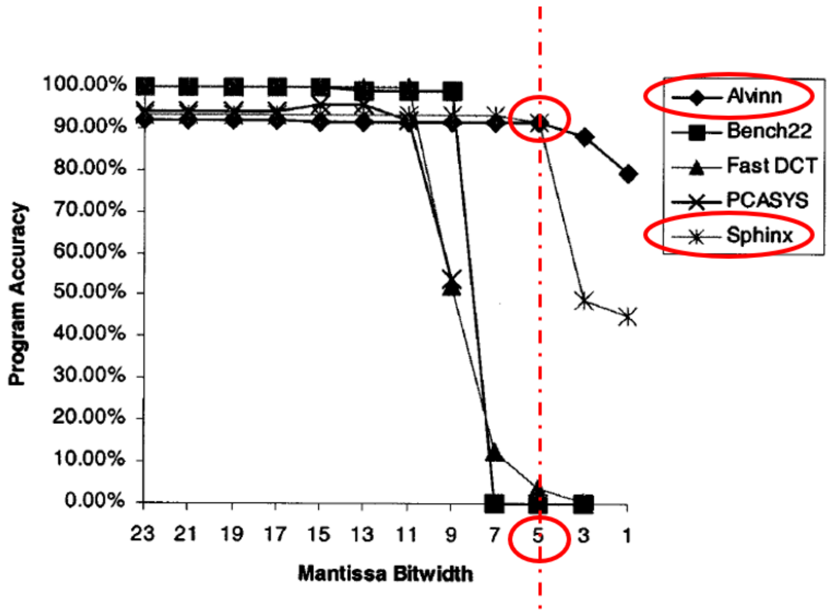


Fig. 6. Program accuracy across various mantissa bitwidths.

Tong, Fig. 6 (annotated)

Having empirically determined the minimum number of mantissa bits necessary to maintain acceptable accuracy of particular applications, Tong teaches using "reduced-precision arithmetic" to "omit the unnecessary bits and computations on them" (Tong, 284), to thereby "reduce waste" (Tong, 273) and "significantly reduce… power consumption while maintaining a program's overall accuracy" (Tong, 274). Tong, 279 (floating-point "provides essential dynamic range… but the fine precision… is not essential"); Goodin, ¶ 320.

B. Claims 1-2 and 16

Tong's teachings and empirical tests would have motivated a POSA to configure Dockser's FPP to operate at the precision levels Tong teaches for particular applications. Goodin, ¶ 321. Tong teaches "[m]any mobile/portable electronics applications" perform signal processing programs "such as speech and image recognition" that are "better served with a custom, reduced[-precision] FP format" (Tong, 284), and a POSA would have understood Dockser's FPP with its selectable precision level for "[p]ower management" in "battery operated devices" (Dockser, [0003]) to be well-suited for use in Tong's mobile/portable applications. Goodin, ¶ 321.

Tong's teaching that a precision level retaining 5 mantissa fraction bits is sufficient in some applications (including ALVINN and Sphinx—*supra* § VI.A) would have motivated a POSA to configure Dockser's FPP in the device discussed *supra* § V to operate at a selected precision level retaining as few as 5 mantissa fraction bits to conserve power when running those applications, or others empirically determined (using Tong's techniques) to not require greater precision. Goodin, ¶ 322. The resulting "Dockser/Tong" device meets claims 1-2 and 16 for the same reasons discussed *supra* § V (Ground 1), with the addition that Dockser/Tong's device retaining 5 fraction bits exceeds claim 1's X and Y values (X=5%, Y=0.05%) by greater margins than Dockser's device retaining 9 fraction bits in Ground 1. Goodin, ¶ 323.

Mr. Goodin's software program (discussed *supra* §§ V.B.4.c(1)(i), V.B.4.c(2) and *infra* Appendix I.B, I.D) that tests all possible valid input pairs demonstrates that when retaining 5 mantissa fraction bits in view of Tong, Dockser's register bit-dropping produces at least Y=0.05% relative error for 99.45% of possible valid inputs, and Dockser's logic bit-dropping produces at least Y=0.05% relative error for 99.47% of possible valid inputs. Goodin, ¶ 324. The algebraic analysis discussed *supra* § V.B.4.c(1)(ii) and *infra* Appendix I.C also demonstrates that Dockser's register bit-dropping when retaining 5 mantissa fraction bits produces a minimum of 1.56% relative error (greater than Y=0.05%) for over 12% of possible valid inputs (greater than X=5%). Goodin, ¶ 324.

Moreover, Tong and Dockser both demonstrate that POSAs knew how to optimize the number of mantissa bits as a result-effective variable for concurrently reducing precision and power consumption in a floating-point execution unit, and how to empirically determine the optimum number of bits for a particular application by testing the application's accuracy at various mantissa bitwidths. Goodin, ¶ 325; *supra* §§ V.B.4.c, VI.A; *infra* § VII.E. The '156 patent lists fifty-six possible combinations of six X percentages (ranging from 1% to 50%) and nine Y percentages (ranging from 0.05% to 20%) as "merely examples [that] do not constitute limitations of the…invention," with no indication that any particular claimed X-Y combination is critical in any way. '156 patent, 27:29-51; Goodin, ¶ 326. Determining the optimum range of imprecision to achieve the best power reduction without sacrificing accuracy for a particular application was a matter of routine optimization of a result-effective variable, such that arriving at imprecisions resulting in the device meeting the claimed X and Y values would have been obvious. Goodin, ¶ 327; *E.I. DuPont de Nemours & Co. v. Synvina C.V.*, 904 F.3d 996, 1010 (Fed. Cir. 2018) ("[D]iscovery of an optimum value of a result effective variable in a known process is ordinarily within the skill of the art."); *Genentech, Inc. v. Hospira, Inc.*, 946 F.3d 1333, 1341 (Fed. Cir. 2020) ("[I]t is not inventive to discover the optimum or workable ranges by routine experimentation."); *In re Urbanski*, 809 F.3d 1237, 1242-1243 (Fed. Cir. 2016) (upholding obviousness finding where there was "no evidence" that "claimed range[]" of result-effective variable was "critical").

C. Claim 33

Tong's teaching to "emulate[] in software different bitwidth FP units" "to determine application accuracy" (Tong, 278) would have motivated a POSA to emulate the Dockser/Tong device (*supra* § VI.B) in software to assess the accuracy of applications running on the device at selected precision levels, and a POSA would have had a reasonable expectation of success in doing so. Goodin, ¶¶ 328-329.

1. [33A1] "A device comprising a computer processor and a computer-readable memory storing computer program instructions, wherein the computer program instructions are executable by the processor to emulate a second device comprising: a plurality of components comprising: at least one first… (LPHDR) execution unit"

When the Dockser/Tong device is "emulated in software" per Tong's teachings (Tong, 278) cited *supra* § VI.A, the conventional and obvious implementation of such "software" is "computer program instructions" stored in "a computer-readable memory" and "executable by [a] processor" within a "device" (*e.g.*, a computer) as claimed. Goodin, ¶ 330; *see* Dockser, [0036] ("software" is "executed by a processor" and "reside[s] in [a] memory"); Donovan (Ex. 1029), [0008]-[0009], [0014] ("software" is "computer program with instructions"); '156 patent, 25:58-60 ("software emulator" is "embodiment[] of the [claimed] invention"). As discussed above (§ VI.C), a POSA would have been motivated by Tong to use a computer (a claimed "device comprising a computer processor…[etc.]") to emulate the Dockser/Tong device discussed *supra* §§ V.D, VI.B (meeting the "second device" of [33A1]), which comprises a plurality of components including a main processor and Dockser's execution unit (which itself comprises multiple components). *Supra* § V.D.1; Goodin, ¶ 331. Like the '156 patent's only examples of "emulat[ing] a… device" comprising LPHDR execution unit(s), Tong emulates the device by executing a software application program that runs thereon with arithmetic operations in the application program replaced by software simulations of the reduced-precision arithmetic the device would perform. '156 patent, 17:5-19, 18:52-19:11; Tong, 278; Goodin, ¶¶ 332-333.

2. Limitations [33A2]-[33B2]

[21A2]-[21B2] are identical to [1A2]-[1B2]. Therefore, because the emulated "second device" in Dockser/Tong is the device discussed *supra* §§ V.B, VI.B, Dockser/Tong meets [33A2]-[33B2] for the same reasons Dockser's device (operating at Dockser's or Tong's disclosed precision levels) meets the identical limitations in claim 1. Goodin, ¶¶ 334-335.

VII. GROUND 3: CLAIMS 1-8 AND 16 WOULD HAVE BEEN OBVIOUS OVER DOCKSER AND MACMILLAN

A. MacMillan

MacMillan (Ex. 1009) "improve[s]… speed of a personal computer architecture through… parallel processing capability" "[t]o provide supercomputer performance in a computer for personal use." MacMillan, 8:38-40, 5:22-45, 1:10-

47. Consistent with how "supercomputer" was used in the art, MacMillan describes supercomputer performance in terms of speed (*e.g.*, number of operations performed per second), not precision. MacMillan, 1:15-54; Ex. 1052, 1411; Ex. 1053, 500; Goodin, ¶¶ 336-337. MacMillan achieves "substantial performance improvements" by adding a "Single Instruction Multiple Data (SIMD)" subsystem to "existing system architectures." MacMillan, 9:17-23, 5:48-6:10. The SIMD subsystem includes SIMD-RAM devices having multiple parallel "processing elements (PE's)," each PE including "floating point accelerators" that "perform… operations on… 32-bit words." MacMillan, 9:11-19, 12:35-59, FIGs. 2 and 5 (reproduced below). Goodin, ¶¶ 338-339.

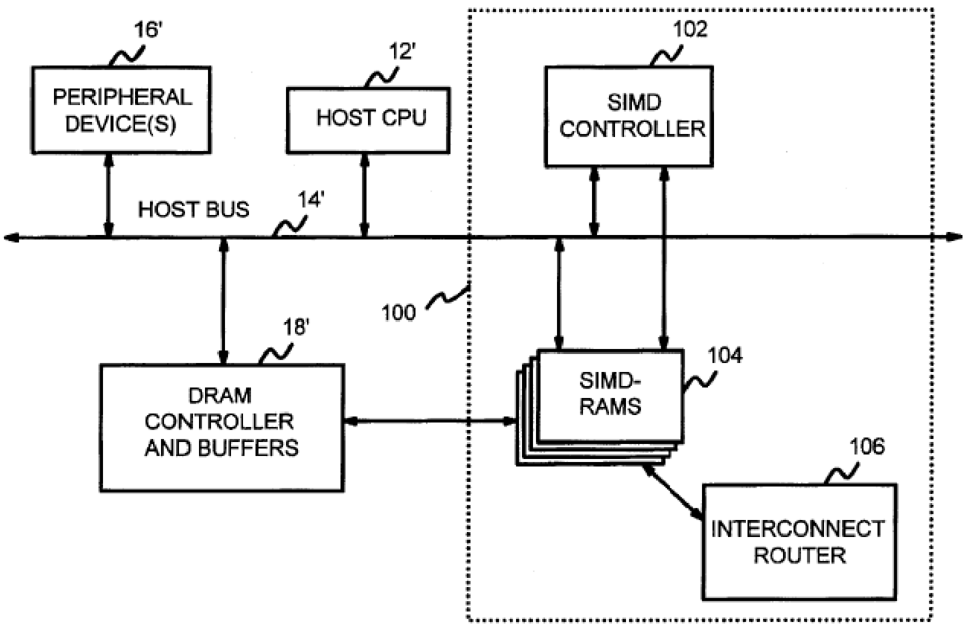


FIG. 2

(SIMD computer system)

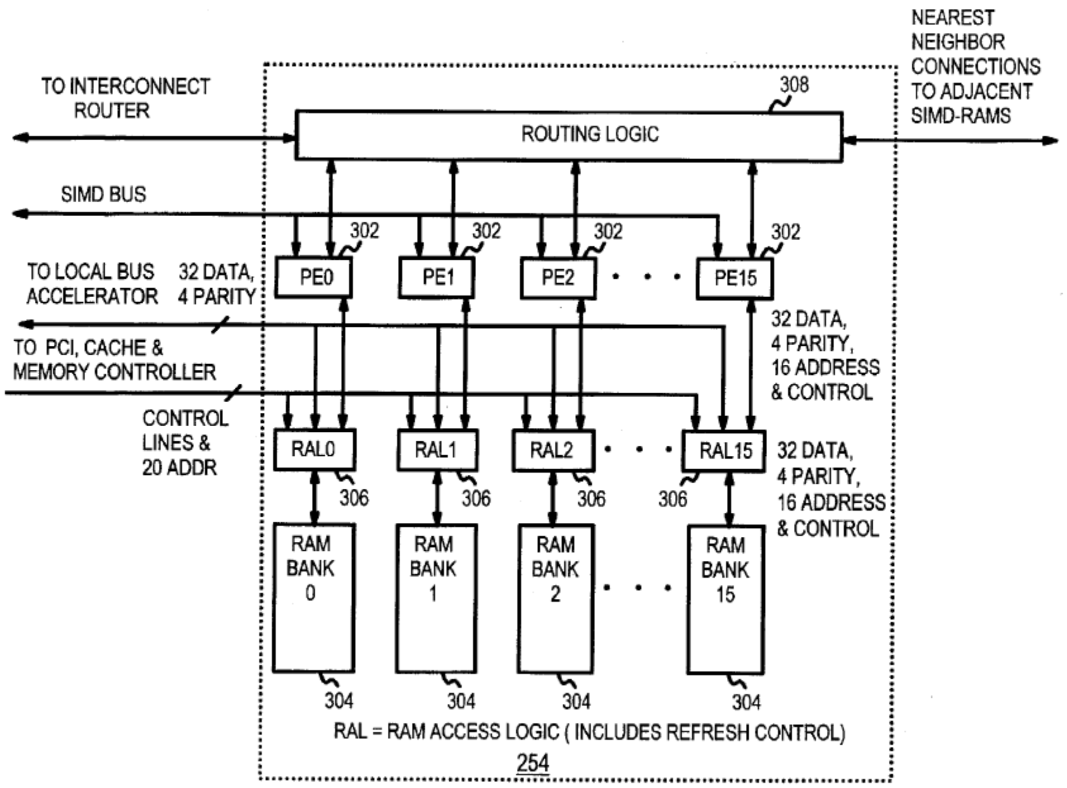


FIG. 5

(SIMD-RAM)

B. Dockser/MacMillan Combination

MacMillan teaches incorporating SIMD parallel processing in personal-use computers, including battery-operated "laptops, palmtops, [and] personal digital assistants"—the same types of computers for which Dockser's FPP provides "[p]ower management" through reduced-precision floating-point operations. MacMillan, 1:6-9, 5:22-45, 7:14-34; Dockser, [0003]; Goodin, ¶ 340. MacMillan teaches "power dissipation and hence power supply capacity and cost" should be considered in pursuing MacMillan's SIMD architecture. MacMillan, 3:2-6; Goodin, ¶ 341. A POSA would have been motivated to use Dockser's FPP to implement each "floating-point accelerator" in the parallel PEs of MacMillan's SIMD architecture to increase performance speed as MacMillan teaches while lowering power consumption as Dockser teaches. Goodin, ¶ 342.

Dockser's FPP is a "floating-point accelerator" for "perform[ing] atomic operations on… 32-bit words" as in MacMillan's PE, and using Dockser's FPP as the floating-point accelerator in MacMillan's PE would have achieved the predictable result of enabling the PEs to perform reduced-precision floating-point arithmetic as taught by Dockser at reduced power. MacMillan, 12:47-59; Goodin, ¶ 343; *see supra* § V.A; Ex. 1024, 1:43-58 ("floating point accelerator" is "processor element" for executing "floating point operation[s]"). MacMillan's SIMD architecture is beneficial for applications including "animation,… rendering…, and video games," which are types of the "graphics applications" for which Dockser teaches its FPP can beneficially save power by reducing unnecessary precision. MacMillan, 7:14-34; Dockser, [0003]; Goodin, ¶ 344.

C. Claims 1-2

In the resulting "Dockser/MacMillan" combination, MacMillan's "computer system" (*e.g.*, 200) (7:14-34, 9:20-29) meets the "device" of [1A1], and comprises Dockser's execution unit) and MacMillan's "Host CPU" (8:56-9:31). Goodin, ¶¶ 345-347. MacMillan's CPU is adapted to control the operation of the SIMD subsystem, including its PEs with Dockser's execution units, by executing a program controlling initiation of "all SIMD processing" (MacMillan, 10:27-53, 13:12-13, 13:38-62); thus the CPU meets [1C] and claim 2 ("CPU"). Goodin, ¶¶ 348-353.

All other limitations of claims 1-2 relate to characteristics of the LPHDR execution unit, which in Dockser/MacMillan is the same Dockser execution unit as in Ground 1. Thus, Dockser/MacMillan meets the remaining limitations in claims 1-2 for the reasons discussed *supra* §§ V.B-V.C. Goodin, ¶ 354.

D. Claim 3: "the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide."

MacMillan's computer system (the claimed "device"—*supra* § VII.C) includes a Host CPU implementable as "a 386, 486 or Pentium™ processor." MacMillan, 9:30-31. POSAs understood these processors included a floating-point execution unit that performed multiplication on IEEE-754 single-format 32-bit floating-point numbers. Goodin, ¶¶ 355-359 (citing Ex. 1017, [0005]; Ex. 1018, 2:7-10; Ex. 1029, [0013]; Ex. 1030, 1:13-19; Dockser, [0002]). Thus, an obvious implementation of MacMillan's CPU includes one claimed "execution unit[]… adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide." Goodin, ¶ 360.

MacMillan discloses an example architecture including 256 PEs, and scaling to "tens of thousands… or more." MacMillan, 12:60-13:4, 13:39-41, 16:20-22, 2:13-16. Thus, in the Dockser/MacMillan combination having a single Host CPU and at least one FPP in each PE (of which there are at least 256), the number of FPPs (LPHDR execution units) exceeds by over 100 the non-negative integer number of Host CPU floating-point execution units. Goodin, ¶¶ 361-362.

The '156 patent says "arithmetic elements… designed to perform… floating point arithmetic with a word length of 32 or more bits" are "designed to perform… arithmetic of traditional precision," and contrasts those traditional-precision elements with "LPHDR arithmetic elements" ('156 patent, 27:52-59), which may "sometimes" produce results that differ from the correct result (*id*., 26:39-49). Goodin, ¶ 363. A POSA would therefore have understood the '156 patent to describe that the claimed "execution units… adapted to execute… multiplication on floating point numbers that are at least 32 bits wide" are "traditional precision" execution units that do not "sometimes" produce results different from the correct traditional-precision result. Goodin, ¶ 364. The floating-point execution unit of MacMillan's Host CPU is such a "traditional precision" execution unit, and Dockser's execution unit that sometimes produces differing results (*supra* § V.B.4) is a claimed "LPHDR execution unit." Goodin, ¶ 365. Thus, a POSA would have understood Dockser/MacMillan meets claim 3 because its number (256 or more) of

Dockser's LPHDR execution units exceeds by over 100 its number (one) of traditional-precision execution units (the single Host CPU floating-point unit). Goodin, ¶ 366.

E. Claims 4-6

Claims 4-6 depend from claim 3 and each recite a minimum X and/or Y percentage higher than claim 1's, the highest combination being "X=10% and Y=0.2%" in claim 6. Mr. Goodin's software program (discussed *supra* §§ V.B.4.c(1)(i), V.B.4.c(2) and *infra* Appendix I.B, I.D) demonstrates that when retaining 9 mantissa fraction bits as Dockser ([0026]) discloses, Dockser's register bit-dropping produces $Y \geq 0.2\%$ relative error for 14.6% of possible valid inputs, and Dockser's logic bit-dropping produces $Y \geq 0.2\%$ relative error for 85% of possible valid inputs. Goodin, ¶ 368. Dockser/MacMillan thus meets claim 6 and claims 4-5 (whose recited X and Y minimum values are no higher than claim 6's). Goodin, ¶¶ 367-368; Appendix I.B-I.D (further including tables showing how each claimed X/Y combination is met).

The algebraic analysis discussed *supra* § V.B.4.c(1)(ii) and *infra* Appendix I.C also demonstrates that Dockser's register bit-dropping meets claim 4 when retaining 9 fraction bits; meets claim 5 when retaining 8 fraction bits; and meets claim 6 when retaining 7 fraction bits. Goodin, ¶ 369. A POSA would have found it obvious to implement Dockser's FPP in Dockser/MacMillan while retaining only

8 or 7 mantissa fraction bits for multiple reasons. First, Dockser's FPP's precision level is "selectable" and a 9-fraction-bit selection is only a non-limiting "example." Dockser, [0003]-[0004], [0026], [0013]; Goodin, ¶ 370. A POSA would have understood Dockser to suggest selecting other below-maximum (Dockser claim 1) precision levels, and selecting a precision level retaining 8 or 7 fraction bits (only 1-2 fewer than Dockser's 9-bit example) to be an obvious implementation of Dockser's teachings. Goodin, ¶ 371. Second, other prior art (*e.g.*, Tong) taught a low-precision floating-point multiplier that retained even fewer fraction bits. Goodin, ¶ 372; *supra* § VI.A. Third, a POSA would have understood the number of mantissa fraction bits retained in Dockser to be a result-effective variable, so that choosing to retain any number would have been obvious in view of Dockser's teachings that the number of mantissa bits dropped/maintained was a "selectable" variable optimizable for different applications, and that the selection impacts the FPP's results through the precision of the outputs the FPP produces. Dockser, [0002]-[0003], [0014], [0024]-[0027]; Goodin, ¶¶ 373-376; *see* result-effective variable caselaw cited *supra* § VI.B.

F. Claim 7

The dynamic range of possible valid inputs to Dockser's operation extends from approximately 2^{-126} (much smaller than 1/1,000,000) to approximately 2^{127}

(much larger than 1,000,000), meeting claim 7. *Supra* § V.B.3; Goodin, ¶¶ 377-378.

G. Claim 8

Dockser's claimed "first operation" is reduced-precision "multiplication," meeting claim 8. *Supra* § V.B.2; Goodin, ¶¶ 379-380.

H. Claim 16

In Dockser/MacMillan, MacMillan's "computer system 200" (7:14-34, 9:20-29) meets the "device" of [16A1], and comprises multiple PEs each including one or more of Dockser's execution unit. Goodin, ¶ 381; *supra* §§ VII.B-VII.C. The PEs are implemented on "SIMD-RAM… chips," and an example architecture includes sixteen SIMD-RAM chips each having sixteen PEs. MacMillan, 12:35-13:4. If claim 16 were interpreted to require "a plurality of components," ***each*** comprising at least one LPHDR execution unit, that plurality of components is met by any two or more of the SIMD-RAM chips, or any two or more of the PEs, in Dockser/MacMillan. Goodin, ¶¶ 382-383.

All other limitations of claim 16 relate to characteristics of the LPHDR execution unit, which in Dockser/MacMillan is the same Dockser execution unit as in Ground 1. Thus, Dockser/MacMillan meets the remaining limitations in claim 16 for the reasons discussed *supra* § V.D. Goodin, ¶ 384.

VIII. GROUND 4: CLAIMS 1-8, 16, AND 33 WOULD HAVE BEEN OBVIOUS OVER DOCKSER, TONG, AND MACMILLAN

A. Claims 1-8 and 16

Dockser/MacMillan (Ground 3) uses Dockser's FPP to implement each "floating-point accelerator" in the parallel PEs of MacMillan's SIMD architecture. *Supra* § VII.B. As discussed in Ground 2, a POSA would have been motivated to implement Dockser's FPP using precision levels as low as 5 mantissa fraction bits, which Tong teaches suffices for applications including "neural network trainer" "ALVINN." Tong, p. 278 & Table IV; *supra* §§ VI.A-VI.B. MacMillan teaches that such "neural net[work]" applications would benefit from "supercomputer performance" provided by using (at least) 256 PEs including "floating point accelerators." MacMillan, 1:24-37, 12:52-13:4; Goodin, ¶¶ 385-386. In view of these combined teachings of Dockser, Tong, and MacMillan, a POSA would have been motivated to use Dockser's FPP with Tong's precision levels as low as 5 mantissa fraction bits as each floating-point accelerator in MacMillan's multiple PEs to achieve "supercomputer performance" for, *e.g.*, neural network applications while conserving power. Goodin, ¶ 387.

The resulting Dockser/Tong/MacMillan combination (*i.e.*, MacMillan's multi-PE system using Dockser's FPP with as low as 5-fraction-bit precision as taught by Tong) meets claims 1-8 and 16 for the same reasons Dockser/MacMillan does discussed *supra* § VII in Ground 3, except the claimed performance characteristics X and Y are met by using 5 rather than 9 mantissa fraction bits, meeting the claimed "at least" X and Y values by the greater amounts explained for Dockser/Tong in Ground 2 *supra* § VI.B. Goodin, ¶ 388.

B. Claim 33

Tong's teachings discussed *supra* §§ VI.A, VI.C to emulate in software a device employing reduced-precision arithmetic would have motivated a POSA to emulate in software the Dockser/Tong/MacMillan device meeting claim 16 (the "second device" recited in claim 33). Goodin, ¶ 389. The computer performing the emulation meets claim 33 for the reasons that same computer does in Ground 2 *supra* § VI.C, except the claimed "second device" being emulated is met by the Dockser/Tong/MacMillan device for the reasons *supra* § VIII.A. Goodin, ¶ 390.

C. Alternative Interpretation of Claim 3

One MacMillan embodiment is "a system… designed primarily… for a specific application, including embedded applications… including… signal processing,… voice recognition, [etc.]." MacMillan, 7:15-34. Tong teaches that many "embedded applications such as voice recognition,… and other human-sensory-oriented signal-processing applications" (Tong, 274), "can get by with less precision/range than full FP" (Tong, 273). Tong experimentally demonstrates that the optimum precision for a "benchmark suite" (Tong, 284) of "five signal processing applications" spanning a "range in complexity" is between 5 and 11 mantissa fraction bits (Tong, 278-279). Goodin, ¶¶ 391-393. A POSA would have been motivated to use Dockser's FPPs in MacMillan's architecture with Tong's precision levels for the reasons *supra* § VIII.A, and to do so in MacMillan's embedded signal-processing system based on Tong's teachings that reduced-precision arithmetic is beneficial in such systems. Goodin, ¶ 394.

In such an embedded system designed specifically for signal processing, a POSA would have been motivated to customize Dockser's FPPs in MacMillan's PEs to only operate at precision levels lower than full FP 32-bit operations, in view of Tong's teachings that "the fine precision of the 23-bit mantissa is not essential." Tong, 279; Goodin, ¶ 395. Dockser ([0017]) teaches that the registers in embodiments of its FPP can be "formatted differently from IEEE 32-bit single format," and a POSA would have been motivated to implement Dockser's FPPs in the embedded signal-processing system with smaller than 32-bit registers to not waste circuit space or incur unnecessary cost in having some register elements that will always be unpowered because they correspond to mantissa bits that will always be tied to "0" in an application-specific system that always operates at reduced precision. Goodin, ¶ 396. Likewise, a POSA would have been motivated to implement the multiplier logic in Dockser's FPP to have only as many logic elements as needed to multiply mantissas of the reduced bitwidth (smaller than 23-

bit) corresponding to the precision level selected for the embedded application. Goodin, ¶ 397.

In this implementation, the claimed "input signal" to Dockser's FPP remains in IEEE-754 32-bit single-format given that the Host CPU and data buses in the Dockser/Tong/MacMillan device use and send to Dockser's FPP standard 32-bit floating-point numbers. MacMillan, 9:30-57; *supra* §§ VII.A, VII.D; Goodin, ¶ 398. However, one or more least-significant mantissa bits of the input signal number are not stored in the register which is implemented with fewer than 32 storage elements. Goodin, ¶ 398. Because the input signal and selected precision levels are unchanged from Grounds 1-3, this implementation of Dockser/Tong/MacMillan meets independent claim 1 (including its "wherein" clause) the same way as in Grounds 1-3 where Dockser's FPP is a claimed LPHDR execution unit. Goodin, ¶ 399; *supra* §§ V-VII.

If dependent claim 3 were interpreted differently than the specification discusses (*see supra* § VII.D), such that an execution unit used for reduced-precision operations could be considered to also meet the claimed "execution unit[]… adapted to execute… multiplication on floating point numbers… 32 bits wide" if its hardware is capable of 32-bit multiplication in some configurations (*e.g.*, if power were applied to all register and logic elements), Dockser's FPP in the above-discussed Dockser/Tong/MacMillan implementation for an embedded application is not that type of execution unit, because its hardware (having smaller-than-32-bit registers and multiplier logic) is ***not*** capable of 32-bit multiplication. Goodin, ¶ 400. Thus, this Dockser/Tong/MacMillan implementation meets that alternative interpretation of claim 3 because only the Host CPU floating-point unit is a claimed "execution unit[]… adapted to execute… multiplication on floating point numbers… 32 bits wide," and the device includes over 100 more of Dockser's FPP (LPHDR execution unit). Goodin, ¶ 401; *supra* § VII.D.

IX. NO BASIS EXISTS FOR DISCRETIONARY DENIAL

A. Section 314(a): Parallel Litigation Does Not Weigh Against Institution

Singular served Petitioner with a complaint on December 20, 2019, and subsequently amended that complaint on March 20, 2020 (Ex. 1032). Petitioner moved to dismiss the amended complaint, which was denied. Exs. 1034, 1036.

Petitioner filed this Petition expeditiously, roughly four months after denial of its motion to dismiss, two months after receiving Singular's infringement contentions, and ***before*** preliminary invalidity contentions are due. Exs. 1033, 1037-1038. *HP Inc. v. Neodron Ltd.*, IPR2020-00459, Paper 17, 40 (Sept. 14, 2020) (instituting trial where petitioner "acted diligently" in filing petitions "approximately two months after" infringement contentions and "two weeks prior" to invalidity contentions); *IBM Corp. v. Trusted Knight Corp.*, IPR2020-00323, Paper 15, 12 (July 10, 2020) (instituting trial where petition filed "soon after" invalidity contentions); *Apple Inc. v. Parus Holdings, Inc.*, IPR2020-00687, Paper 9, 16 (Sept. 23, 2020) (instituting trial where petition filed "shortly after…preliminary invalidity contentions").

The factors in *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential) ("*Fintiv*") weigh against discretionary denial.

1. Factor-1: Potential for Stay

If instituted, Petitioner will move to stay the litigation, but without "specific evidence" of how the Court will rule this factor is neutral. *Sand Revolution II v. Continental Intermodal Group–Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative).

2. Factor-2: Trial Timing

No trial date has been set, Ex. 1038, which "***weighs heavily against denying institution***." *IBM*, IPR2020-00323, Paper 15, 11. The district court was clear: "this case is…very far away from trial." Ex. 1039, 5:4-7.

The impact of the "global pandemic on court congestion and trial dates is likely to add delay." *Google LLC v. Uniloc 2017 LLC*, IPR2020-00479, Paper 10, 12 (Aug. 13, 2020). Civil jury trials in the District of Massachusetts are continued pending further order, Ex. 1040, and "it may be some time before [the district court is] even in a position ***to think about scheduling*** a trial…." Ex. 1039, 5:4-7.

3. Factor-3: Investment in Litigation

The parties have thus far made no investment related to section 102 or 103 issues as Petitioner's preliminary invalidity contentions are not yet due, and expert reports will not be exchanged until well after an institution decision is due. Ex. 1038. Factor-3 thus weighs against institution denial. *Apple*, IPR2020-00687, Paper 9, 17-19 (explaining factor-3 focuses on investment in issues the Board has authority to address and weighs against denial where expert discovery "has yet to take place"); *Sand Revolution*, IPR2019-01393, Paper 24, 10-11 (similar).

Fact discovery is also in its infancy with minimal discovery and no depositions. *HP*, IPR2020-00459, Paper 17, 40 ("fact discovery in its infancy" weights against denial); *NVIDIA Corp. v. Invensas Corp.*, IPR2020-00602, Paper 11, 27 (Sept. 3, 2020) (factor-3 weighs against denial where "significant work remains"). Moreover, as explained above, Petitioner filed this petition diligently, which "mitigates against" any investment in the litigation. *HP*, IPR2020-00459, Paper 17, 40; *see also IBM*, IPR2020-00323, Paper 15, 12; *Apple*, IPR2020-00687, Paper 9, 16.

4. Factor-4: Overlap of Issues

Although specific validity issues to be raised in the litigation are largely unknown given its early stages, the Board nonetheless will decide many issues that do not overlap with litigation issues because Petitioner challenges nine claims not asserted in the litigation. *Apple Inc. v. Maxell, Ltd.*, IPR2020-00200, Paper 11, 17 (July 15, 2020) (challenging unasserted claims weighs in favor of institution).

Failure to consider patentability of the unasserted claims would prejudice Petitioner, who would be barred from challenging those additional claims in a future IPR should Singular later assert those claims in the pending litigation (or some future litigation). Singular originally asserted four claims against Petitioner (Ex. 1032, ¶ 104), subsequently provided infringement contentions for only one of those four (Ex. 1037), and has expressly reserved the right to re-assert the previously-asserted claims and/or newly assert additional claims (Ex. 1041, 2).

5. Factor-5: Petitioner Is the Litigation Defendant

Petitioner is the defendant, but factor-5 weighs against denial because the Board is likely to reach a final decision before the district court trial. *Supra* § IX.A.2; *Google*, IPR2020-00479, Paper 10, 18.

6. Factor-6: Other Circumstances, Including Merits

This Petition shows the challenged claims are demonstrably unpatentable, which weighs against denial of institution. *Fintiv*, 14-16.

B. <u>Section 325(d)</u>: The Petition Presents New Art and Arguments Not Previously Considered

In applying § 325(d), the Board uses the two-part framework of *Advanced Bionics v. MED-El Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 8-9 (Feb. 13, 2020) (precedential).

The Petition's grounds are based on references neither considered during prosecution nor duplicative of art previously considered. *Supra* § IV.D.

Thus "the first part of the framework set forth in *Advanced Bionics* is not met," and the Board "need not reach" its second prong. *TalexMedical, LLC v. Becon Med., Ltd.*, IPR2020-00030, Paper 7 at 9-10 (Apr. 17, 2020) (first prong not met where petition's references were not of record and no rejections or substantive arguments were made during prosecution).

X. CONCLUSION

The Board should institute review and cancel claims 1-8, 16, and 33.

Dated: November 5, 2020

Respectfully submitted,
Google LLC

By__/Elisabeth Hunt/__________
Elisabeth H. Hunt, Reg. No. 67,336
WOLF GREENFIELD & SACKS, P.C.

XI. APPENDIX I

A. Multiplier Relative Error Independent of Exponent/Sign

Given an input pair of floating-point numbers A and B with sign bits S_A and S_B, exponents E_A and E_B, and mantissas M_A and M_B, the exact mathematical calculation of the product $Q = A \times B$ is:

$$Q = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times (M_A \times M_B)$$

where '$\otimes$' denotes an XOR operation. Goodin, ¶¶ 402-405.

Representing the mantissa product $(M_A \times M_B)$ as V yields:

$$Q = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V$$

Goodin, ¶ 406.

Both Dockser's register bit-dropping and logic bit-dropping techniques alter the product's mantissa but not its sign or exponent. *Supra* § V.B.4.c; Goodin, ¶ 407. Letting V' be the altered mantissa product, the product Q' of Dockser's reduced-precision multiplication is:

$$Q' = (-1)^{(S_A \otimes S_B)} \times 2^{(E_A + E_B)} \times V'$$

Goodin, ¶ 408.

The claimed relative error percentage Y is:

$$Y = \left| \frac{Q - Q'}{Q} \right| \times 100$$

Goodin, ¶ 409; '156 patent, 26:44-60.

Substituting the above expressions for Q and Q' yields:

$$Y = \left| \frac{\left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A+E_B)} \times V\right) - \left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A+E_B)} \times V'\right)}{(-1)^{(S_A \otimes S_B)} \times 2^{(E_A+E_B)} \times V} \right| \times 100$$

Goodin, ¶ 410.

Factoring out the exponent and sign terms yields:

$$Y = \left| \frac{\left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A+E_B)}\right) \times (V - V')}{\left((-1)^{(S_A \otimes S_B)} \times 2^{(E_A+E_B)}\right) \times V} \right| \times 100$$

Goodin, ¶ 411.

The exponent and sign terms in the numerator and denominator cancel out, yielding:

$$Y = \left| \frac{V - V'}{V} \right| \times 100$$

Goodin, ¶ 412.

Thus, the claimed "Y" percentage produced by Dockser's reduced-precision multiplication depends only on the operand mantissas and is independent of the exponents and signs. Goodin, ¶ 413.

B. Software Demonstration of Register Bit-Dropping

Mr. Goodin's software program for Dockser's register bit-dropping iterates through all possible pairs of normal IEEE-754 single-format floating-point numbers A and B that are non-negative (having '0' sign bit) with zero-valued exponent (such that the mantissa is multiplied by $2^0 = 1$), and for each pair:

1. Performs the exact mathematical calculation of $Q = A \times B$;

2. Retains a specified number of mantissa fraction bits for A and B and zeroes the remaining less-significant fraction bits to create reduced-mantissa operands A' and B';

3. Computes the reduced-precision product $Q' = A' \times B'$; and

4. Computes the relative error percentage $Y = \left|\frac{Q-Q'}{Q}\right| \times 100$.

The program counts the number of input pairs for which Y is at least the minimum percentage recited in a challenged claim. Goodin, ¶¶ 414-424.

Testing only non-negative single-format floating-point numbers with zero-valued exponent is sufficient to test all possible pairs of single-format operands, because the relative error percentage produced by Dockser's multiplication is independent of operand exponent and sign, and the set of all possible single-format operands combines all possible mantissas with all possible exponents and signs. Goodin, ¶ 421; *supra* Appendix I.A.

The results below from Mr. Goodin's program demonstrate that Dockser's register bit-dropping technique produces the claimed minimum relative error percentage Y for more than the claimed minimum percentage X of the possible valid inputs at precision levels retaining 9 (Dockser, [0026]) or 5 (Tong, 282) fraction bits in the operands for each claimed X/Y combination. The reported "Dockser's X" percentages conservatively exclude pairs that could produce overflow/underflow exceptions.

Retained Fraction Bits	Claimed Y%	Dockser's X%
9	≥ 0.05%	≥ 92.14333%
	≥ 0.20%	≥ 14.59791%
5	≥ 0.05%	≥ 99.44834%
	≥ 0.20%	≥ 99.02600%

Goodin, ¶¶ 425, 468-473.

A POSA would have understood the claimed "exact mathematical calculation" of $Q = A \times B$ can have 48 bits in its mantissa (the product of two 24-bit mantissas). Goodin, ¶ 426; Dockser, [0034], Fig. 3B. Mr. Goodin's program therefore stores the exact product in IEEE-754 double-precision floating-point format. If Singular were to argue that the "exact mathematical calculation" of multiplication of two single-format floating-point operands is the product rounded or truncated to single format (23-bit-fraction mantissa), the claims are still met because Mr. Goodin's test using this alternative produced identical results up to four decimal places of the X percentage. Goodin, ¶¶ 427-428.

Retained Fraction Bits	Claimed Y%	Dockser's X%
9	≥ 0.05%	≥ 92.14333%
	≥ 0.20%	≥ 14.59792%
5	≥ 0.05%	≥ 99.44834%
	≥ 0.20%	≥ 99.02600%

C. Algebraic Analysis of Register Bit-Dropping

For single-format floating-point operands A and B with sign bits S_A and S_B, exponents E_A and E_B, and mantissas M_A and M_B, respectively, the exact mathematical calculation of multiplying mantissas M_A and M_B is:

$$V = M_A \times M_B$$

Dockser's register bit-dropping technique drops least-significant bits from (truncates) the mantissa of each operand. *See supra* § V.B.4.c(1). Letting M_A' and M_B' be the truncated versions of mantissas M_A and M_B, and V' be the product of M_A' and M_B' that Dockser's register bit-dropping technique produces,

$$V' = M'_A \times M'_B$$

The claimed relative error percentage "Y" is:

$$Y = \left| \frac{V - V'}{V} \right| \times 100$$

Goodin, ¶¶ 429-430; *supra* Appendix I.A.

Letting D_A be the difference between M_A and M_A', and letting D_B be the difference between M_B and M_B', *i.e.*:

$$M'_A = M_A - D_A$$

$$M'_B = M_B - D_B$$

The output product of Dockser's register bit-dropping technique is then:

$$V' = M_A' \times M_B' = (M_A - D_A) \times (M_B - D_B)$$

Goodin, ¶ 431.

Substituting the above expressions for V and V' into the expression for Y yields:

$$Y = \left|\frac{(M_A \times M_B) - ((M_A - D_A) \times (M_B - D_B))}{(M_A \times M_B)}\right| \times 100\%$$

Goodin, ¶ 432.

Expanding the numerator to show all individual products yields:

$$Y = \left|\frac{\big((M_A \times M_B) - (M_A \times M_B) + (D_A \times M_B) + (M_A \times D_B) - (D_A \times D_B)\big)}{(M_A \times M_B)}\right| \times 100\%$$

In the numerator, $(M_A \times M_B) - (M_A \times M_B)$ cancels out, yielding:

$$Y = \left|\frac{(D_A \times M_B) + (M_A \times D_B) - (D_A \times D_B)}{(M_A \times M_B)}\right| \times 100\%$$

The fraction can be re-written as the sum of three fractions:

$$Y = \left|\frac{D_A}{M_A} + \frac{D_B}{M_B} - \frac{(D_A \times D_B)}{(M_A \times M_B)}\right| \times 100\% \qquad (\boldsymbol{Equation\ A})$$

Goodin, ¶¶ 433-435.

M_A and M_B are within the range $1 \le mantissa < 2$ (*see supra* § V.A); therefore $(M_A \times M_B) \ge M_B$. Goodin, ¶ 436. Because D_A and D_B are fractional differences between an original mantissa and its truncated version, D_A and D_B are both less than 1; therefore $(D_A \times D_B) < D_B$. Goodin, ¶ 437. Therefore:

$$\frac{D_B}{M_B} \ge \frac{(D_A \times D_B)}{(M_A \times M_B)}$$

(because the right-hand fraction has a smaller numerator and a larger denominator than the left-hand fraction), and thus the quantity $\frac{D_B}{M_B} - \frac{(D_A \times D_B)}{(M_A \times M_B)}$ in Equation A is a positive number. Goodin, ¶ 438. Thus, the relative error percentage is at least as large as the first fraction in Equation A; *i.e.*:

$$Y \geq \left|\frac{D_A}{M_A}\right| \times 100\% \qquad (\textbf{\textit{Equation B}})$$

Goodin, ¶ 439. The ratio $\frac{D_A}{M_A}$ expressed as a percentage therefore provides a lower bound on the relative error percentage resulting from Dockser's multiplication with register bit-dropping on any two operands.

50% of all possible values of M_A have a zero as the first (most-significant) fraction bit (*i.e.*, M_A= 1.0…); the other 50% have a one as the first bit (*i.e.*, M_A= 1.1…). Goodin, ¶ 440.

The value of D_A (the difference between the full and truncated mantissas of operand A) is determined by the $(K+1)^{th}$ through the 23rd fraction bits of M_A, where K is the number of fraction bits to which M_A is truncated in Dockser's register bit-dropping; *e.g.*, when K=9 (*see* Dockser, [0026]), M_A's 10th through 23rd fraction bits are zeroed. Goodin, ¶ 441.

Of the 50% of M_A values that have a first fraction bit of zero, 25% have ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits. Goodin, ¶ 442. In this 12.5% of all possible values of M_A (25% of 50%), the value of D_A is at least $\big(2^{-(K+1)} +$

$2^{-(K+2)}\big)$, which is the difference produced when ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits are changed to zeros; and the value of M_A is no larger than $(1.5 - 2^{-23})$, which is the largest possible mantissa having a zero as the first fraction bit (*i.e.*, 1.0111…). Goodin, ¶ 443.

Referring to Equation B, therefore, the following inequality holds for the 12.5% of all possible M_A values with zero in the first fraction bit and ones in the $(K+1)^{th}$ and $(K+2)^{th}$ fraction bits, when Dockser's register bit-dropping truncates the operands to K fraction bits:

$$Y \geq \frac{\big(2^{-(K+1)} + 2^{-(K+2)}\big)}{(1.5 - 2^{-23})} \times 100\% \qquad (\boldsymbol{Equation\ C})$$

Goodin, ¶ 444. Since each M_A value can be paired with every possible value of M_B, the above Equation C is also true for 12.5% of all possible ***pairs*** of mantissas M_A and M_B; thus, for a given subprecision retaining K fraction bits with Dockser's register bit-dropping technique, over 12% of all possible valid pairs of input operands will produce ***at minimum*** the relative error given by Equation C. Goodin, ¶ 445.

Similarly, 25% of all possible values of M_A have zeros as the first ***two*** fraction bits (*i.e.*, M_A= 1.00…), and 25% of those have ones in both the $(K+1)^{th}$ and $(K+2)^{th}$ bits. Goodin, ¶ 446. In this 6.25% of all possible values of M_A (25% of 25%), the value of D_A is at least $\big(2^{-(K+1)} + 2^{-(K+2)}\big)$, and the value of M_A

is no larger than $(1.25 - 2^{-23})$, which is the largest possible mantissa having zeros as the first two fraction bits (*i.e.*, 1.00111…); therefore, the following inequality holds for the 6.25% of all possible input pairs in which M_A has zeros in the first two fraction bits and ones in the $(K+1)^{th}$ and $(K+2)^{th}$ fraction bits, when Dockser's register bit-dropping truncates the operands to K fraction bits:

$$Y \geq \frac{\left(2^{-(K+1)} + 2^{-(K+2)}\right)}{(1.25 - 2^{-23})} \times 100\% \qquad (\boldsymbol{Equation\ D})$$

Goodin, ¶ 446. Thus, for a given subprecision retaining K fraction bits with Dockser's register bit-dropping technique, over 6% of all possible valid pairs of input operands will produce ***at minimum*** the percent error given by Equation D. Goodin, ¶ 447.

The table below provides the results of evaluating Equations C and D with various values K of retained fraction bits as the selected precision level.

Retained Fraction Bits (K)	**Equation C: Minimum Y for $X \geq 12\%$**	**Equation D: Minimum Y for $X \geq 6\%$**	**Meets X/Y Percentages Recited by Claims:**
9	$\geq 0.0976\%$	$\geq 0.1171\%$	1, 4, 16, 33
8	$\geq 0.1953\%$	$\geq 0.2343\%$	All above plus 5
7	$\geq 0.3906\%$	$\geq 0.4687\%$	All above plus 6
5	$\geq 1.5625\%$	$\geq 1.8750\%$	All above

Goodin, ¶ 448.

D. Logic Bit-Dropping

Mr. Goodin's software program for Dockser's logic bit-dropping iterates through all possible pairs of non-negative ('0' sign bit) normal IEEE-754 single-format floating-point numbers A and B with zero-valued exponent (mantissa is multiplied by $2^0 = 1$), and for each pair:

1. Performs the exact mathematical calculation of $Q = A \times B$;

2. Performs Dockser's logic bit-dropping to compute Q' from A and B, as explained below; and

3. Computes the relative error percentage $Y = \left|\frac{Q-Q'}{Q}\right| \times 100$.

The program counts the number of input pairs for which Y is at least the minimum percentage recited in a challenged claim. Goodin, ¶¶ 449-454.

Testing all possible pairs of non-negative single-format floating-point numbers with zero-valued exponent suffices for the reasons discussed *supra* Appendix I.B. Goodin, ¶ 455.

Mr. Goodin's program computes Q' (step 2 above) using the following technique (Dockser, [0031]-[0034], Fig. 3B):

1. Initialize the "output value" (Dockser, [0031]) of the mantissa product to 0;
2. Interpret the bit sequences of the mantissas of operands A and B (including the implied leftmost "1" bit) as 24-bit integers representing the "multiplicand 402" and "multiplier 404" (Dockser, [0030]-[0031]);

3. For each bit in the multiplier,
    a. compute a partial product that equals 0 (when the multiplier bit is 0) or the multiplicand (when the multiplier bit is 1) (Dockser, [0031]);
    b. left-shift the partial product by inserting a number of 0s, at the partial product's right end, equal to the multiplier bit's bit position (Dockser, [0031]);
    c. Convert to 0 all partial product bits less significant than the "selected subprecision" (Dockser, [0032]-[0033], Fig. 3B); and
    d. Add the result of step c to the output value (Dockser, [0031], [0034]).
    e. (Repeat step 3 for each bit in the multiplier.)
4. Interpret the generated sum as a binary output number Q' in which the radix point is to the left of the rightmost 46 bits of the generated sum.

Goodin, ¶¶ 456-464.

The results below from Mr. Goodin's program, conservatively excluding pairs that could produce overflow/underflow exceptions from the reported X percentage, demonstrate that Dockser's logic bit-dropping produces the claimed minimum relative error percentage Y for more than the claimed minimum percentage X of the possible valid inputs at precision levels retaining 9 (Dockser, [0026]) or 5 (Tong, 282) fraction bits in the output for each claimed X/Y combination.

Retained Fraction Bits	Claimed Y%	Dockser's X%
9 fraction bits	≥ 0.05%	≥ 99.35080%
	≥ 0.20%	≥ 84.99052%
5 fraction bits	≥ 0.05%	≥ 99.47388%
	≥ 0.20%	≥ 99.43546%

Goodin, ¶¶ 465-473.

The program stores the exact product in IEEE-754 double-precision floating-point format. *See supra* Appendix I.B. If Singular were to argue that the "exact mathematical calculation" of multiplying two single-format floating-point operands is a single-format product (23-bit-fraction mantissa), the claims would still be met because Mr. Goodin's test using this alternative produced results identical to the above table up to five decimal places of the X percentage. Goodin, ¶ 467.

Retained Fraction Bits	Claimed Y%	Docksers's X%
9 fraction bits	≥ 0.05%	≥ 99.35080%
	≥ 0.20%	≥ 84.99052%
5 fraction bits	≥ 0.05%	≥ 99.47388%
	≥ 0.20%	≥ 99.43546%

XII. APPENDIX II: CLAIM LISTING

The following claim listing assigns element labels (e.g., [1A1], [1A2], etc.) to certain claims for clarity.

Claim 1
[1A1] A device comprising: at least one first low precision high dynamic range (LPHDR) execution unit
[1A2] adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
[1B1] wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and
[1B2] for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input; and
[1C] at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit.
Claim 2
The device of claim 1, wherein the at least one first computing device comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine.
Claim 3
The device of claim 2, wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.
Claim 4
The device of claim 3, wherein X=10%.
Claim 5
The device of claim 3, wherein Y=0.2%.
Claim 6
The device of claim 3, wherein X=10% and Y=0.2%.

Claim 7
The device of claim 3, wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.
Claim 8
The device of claim 3, wherein the first operation is multiplication.
Claim 16
[16A1] A device comprising: a plurality of components comprising: at least one first low precision high dynamic range (LPHDR) execution unit
[16A2] adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
[16B1] wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and
[16B2] for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input.
Claim 33
[33A1] A device comprising a computer processor and a computer-readable memory storing computer program instructions, wherein the computer program instructions are executable by the processor to emulate a second device comprising: a plurality of components comprising: at least one first low precision high-dynamic range (LPHDR) execution unit
[33A2] adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value;
[33B1] wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/65,000 through 65,000 and
[33B2] for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X% of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input.

CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (e)(4)

I certify that on November 5, 2020, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via Overnight FedEx at the following correspondence address of record for the patent:

Blueshift IP LLC
1 Broadway, 14th Floor
Cambridge, MA 02142

Date: November 5, 2020

/MacAulay Rush /
MacAulay Rush
Paralegal
WOLF, GREENFIELD & SACKS, P.C

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,787 words (including the words in Appendix I) excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: November 5, 2020

/Alexandra H. Kime/
Alexandra H. Kime
Paralegal
WOLF, GREENFIELD & SACKS, P.C